PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GERADOR DE SOFTWARE (PUC-GO)

Aluno: Murilo Barros Peixoto

GOIÂNIA
2020

MURILO BARROS PEIXOTO

DESENVOLVIMENTO DO GERADOR DE SOFTWARE

Trabalho de Conclusão de Curso apresentado à Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da computação.

Orientador(a): Prof. Me. Eugênio Júlio M. Candido Carvalho

Banca examinadora:

Prof. Dr. José Luiz de Freitas Júnior
Prof. Dr. Fábio Barbosa Rodrigues

GOIÂNIA
2020

MURILO BARROS PEIXOTO

DESENVOLVIMENTO DO GERADOR DE SOFTWARE

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Ciência de Computação, em _____/_____/_____.

_____
Orientador(a): Me. Eugênio Júlio M. Candido Carvalho

_____
Profa. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

GOIÂNIA
2020

AGRADECIMENTOS

Agradeço primeiramente a Deus, que sempre me deu forças para buscar meus objetivos e meus sonhos e nunca me deixou abaixar a cabeça. Aos meus pais, que sempre me apoiaram e me ajudaram em minhas dificuldades, me dando o suporte necessário para eu me tornar a pessoa que sou hoje. A minha namorada, que mesmo antes de ser minha namorada me ajudava, aconselhava e incentivava a sempre ir além dos meus limites. Ao meu professor Eugenio que desde o início de minha graduação me apoio e acreditou em mim. E por último, mas não menos importante, agradeço minha cachorrinha baby, que sempre me escutava quando eu reclamava e levantava o meu animo em dias ruins.

"Se os construtores construíssem edifícios da maneira como os programadores escrevem programas, o primeiro pica-pau que surgisse destruiria a civilização."

(Gerald Weinberg)

RESUMO

A automatização de processos está presente em diversas áreas e em praticamente todos os lugares, porém, seu uso no desenvolvimento de software ainda é pouco explorado. Este trabalho apresenta o desenvolvimento de um *framework* capaz de desenvolver um software sem a escrita de nenhuma linha de código e auxiliar profissionais de alto e baixo conhecimento em programação a desenvolverem softwares seguros e escaláveis. O software é capaz de criar telas, campos, dados e relacionamentos (1 para 1, N para 1 e N para N) dinâmicos e assim dando vida a um sistema responsável por atender a necessidade do usuário. O *framework* fornece ao usuário a possibilidade de criação de um sistema capaz de realizar as operações de cadastros, alterações, visualizações e exclusão dados, além de, ter sua criação no formato CD, ou seja, toda alteração é refletida em tempo real em ambiente produtivo facilitando assim também a etapa de impalnatação do software. O *framework* foi desenvolvido utilizando o ferramental oferecido pela empresa Microsoft como a linguagem de programação C#, o *framework* .Net Core, Asp.Net Core MVC, banco de dados SQL Server e uma API RestFul.

Palavras-Chaves: *Framework,* C#, .Net Core, Asp.Net Core MVC, SQL Server, API RestFul, CD.

ABSTRACT

Process automation is present in several areas and practically everywhere, however, its use in software development is still little explored. This work presents the development of a framework capable of developing software without writing any line of code and assisting professionals with high and low knowledge in programming to develop secure and scalable software. The software is capable of creating dynamic screens, fields, data and relationships (1 to 1, N to 1 and N to N) and thus giving life to a system responsible for meeting the user's needs. The framework provides the user with the possibility of creating a system capable of performing data registration, changes, views and deletion operations, in addition to having its creation in CD format, that is, any change is reflected in real time in a productive environment. thus also facilitating the software deployment stage. The framework was developed using the tooling offered by Microsoft as the C # programming language, the .Net Core framework, Asp.Net Core MVC, SQL Server database and a RestFul API.

Keywords*: Framework, C#, .Net Core, Asp.Net Core MVC, SQL Server, API RestFul.*

1 LISTA DE FIGURAS

2 LISTA DE QUADROS

Tabela de Conteúdos

1 Introdução

De acordo com (DOS SANTOS,2020) a aceleração contínua das inovações tecnológicas demanda cada vez mais eficiência na produção, pois a competitividade aumenta em virtude das facilidades de conexão entre fronteiras que em um rol de vantagens tem uma que é a facilitação de comércio de produtos.

A produtividade vem aliada ao custo, porém pode ser entendida como um benefício em si, tendo sua mensuração através da comparação do tempo antigo com o novo tempo, comparando recursos novos com antigos. Seja como uma reengenharia de processos, entregando uma nova tecnologia capaz de reorganizar estes processos, seja através de automatização de processos antes manuais ou atividades mais ágeis (Toledo. R, 2020).

A automação ajuda a eliminar as atividades redundantes que geram estresse quando executadas por pessoas e ajuda a criar demanda por mão de obra especializada e isto gera oportunidades de empregos nas quais o ser humano é mais valorizado (DOS SANTOS,2020).

De acordo com (Toledo. R, 2020) este aumento da produtividade, além da velocidade de produção, nos remete à qualidade do que se entrega, seja através de TI no produto em si ou mesmo em etapas do processo, gerando menos erros, mais conformidade.

1.1 Objetivo Geral

Este trabalho tem como objetivo o desenvolvimento de um *framework* para a automatização do processo de desenvolvimento de software junto ao processo de implantação do sistema em ambiente produtivo, com o propósito de auxiliar pessoas com um conhecimento prévio ou até mesmo pessoas com conhecimento avançado a desenvolverem softwares seguros e escaláveis de maneira simples.

1.2 Objetivo Específico

- Levantamento de requisitos;
- Validação e documentação dos requisitos;
- Estudo das tecnologias para construção do *framework*;
- Implementação do sistema para web;
- Implementação do sistema para dispositivos moveis Android.
- Implementação do sistema para *Windows* e *Linux*.
- Validação e teste no sistema.

1.3 Definições, Siglas e Abreviações

API	*Application Programming Interface*
Cacheamento	Aplicação de estratégia de cache
CD	*Continuous Deployment*
Metacampo	Dado lógico representante de um campo
Metadado	Dado lógico representante de uma informação
Metaformulario	Dado lógico representante de uma tela
N/A	Não se aplica
PK	*Primary Key*
PWA	*Progressive Web App*
Tupla	Conjunto de metadados

1.4 Visão Geral

Este trabalho está dividido em cinco partes.

1. Introdução ao trabalho junto a seus objetivos gerais e específicos, definições e justificativa.
2. Descrição geral em relação ao produto desenvolvido, como as interfaces do sistema, usuário, *software*, *hardware* e comunicação.
3. Requisitos de usuários, funcionais, de qualidade, diagrama de caso de uso e caso de uso descritivo.
4. Ferramental utilizado no desenvolvimento do sistema e apresenta os resultados obtidos.

5. Considerações finais e sugestões para trabalhos futuros.

2 Descrição Geral

Este tópico apresenta a descrição geral do sistema, explicando sobre aspectos gerais do produto, tais como interfaces do sistema, do usuário, de hardware, software e comunicação.

2.1 Aspecto Geral do Produto

O Gerador de software é um sistema WEB, *Desktop* e *mobile* com o intuito de auxiliar usuários com pouco ou alto conhecimento de programação a desenvolver um *software* de maneira simples e rápida, além de facilitar a implantação do produto em ambiente produtivo.

A principal finalidade do sistema é proporcionar ao usuario uma forma organizada, simples e segura de desenvolver softwares.

2.2 Interfaces do sistema

A seguir serão apresentadas as descrições das telas do sistema Gerador de Software. O sistema foi divido em dois níveis: Usuário e *Root*.

Para o nível Usuário o sistema possui as seguintes telas:

- ***Login:*** Contém campos de e-mail e senha e uma opção de cadastro no sistema.
- **Cadastro de usuário*:*** Possui o campo e-mail, senha e confirmação de senha, para a realização do cadastro no sistema.
- **Relatório Geral*:*** Possui informações sobre os metaformularios em relação ao seus metadados.
- **Cadastros de metainformação*:*** Possui campos gerados dinamicamente a partir de um metaformulario escolhido com a possibilidade de inserção de dados.

- **Edição de metainformação*:*** Possui campos gerados dinamicamente a partir de um metaformulario escolhido com a possibilidade de edição de dados.
  - **Listagem de metainformação*:*** Possui uma tabela de campos gerados dinamicamente com opções de edição e exclusão.
  - **Exclusão de metainformação*:*** Possui uma opção para excluir uma tupla ou cancelar a exclusão.

O nível *Root* possui todas as telas do nível Usuário com adição das seguintes telas:

- **Cadastro de metaformularios*:*** Contém campos para informar o nome do metaformulario, seu rotulo e seu tipo.
- **Edição de metaformularios*:*** Contém campos para alterar o nome do metaformulario e seu rotulo.
- **Listagem de metaformularios*:*** Lista de metaformularios cadastrados onde cada item da lista possui a ações para edição, exclusão e visualização de metacampos.
- **Exclusão de metaformularios*:*** Possui uma opção para excluir um metaformulario ou cancelar a exclusão.
- **Cadastro de metacampos*:*** Contém um formulário com campos para o cadastro dos metacampos.
- **Edição de metacampos*:*** Contém um formulário com campos para o edição dos metacampos.
- **Listagem de metacampos*:*** Lista de metacampos cadastrados onde cada item da lista possui opções de edição e exclusão.
- **Exclusão de metacampos*:*** Possui uma opção para excluir um metacampo ou cancelar a exclusão.
- **Configuração do sistema:** Possui campos para configuração do nome do sistema, texto de sobre, opção para publicação e para geração de *PWA*.

2.3 Interfaces do usuário

O sistema foi desenvolvido para ser móvel, desktop e WEB, podendo ser utilizado em qualquer *smartphone* que tenha o sistema operacional *Android* e em qualquer computador com os sistemas operacionais *Windows* ou *Linux.*

2.4 Interfaces do hardware

Para que o sistema funcione corretamente os dispositivos devem atender aos seguintes requisitos.

- *Smartphones*
  - Intel Quad Core;
  - *Random Acess Memory* (RAM) de 500mb;
  - Espaço livre de 5mb.
- Desktop e WEB
  - Processador *Dual Core;*
  - RAM de 500mb;
  - Espaço livre de 5mb.

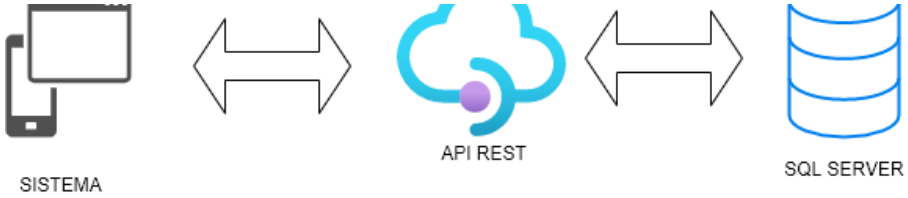
2.5 Interfaces do software

O sistema foi desenvolvido para a versão 7 ou superior do *Windows*, todas as versões de *Linux* e Android 6.0 ou superior.

2.6 Interfaces de comunicação

O sistema foi desenvolvido utilizando uma API RestFul que utiliza o banco de dados SQL Server. A figura 1 representa o modelo de implementação seguido.

Figura 1 – Comunicação entre o sistema e API



Fonte: Elaborado pelo autor.

2.7 Operações

O Gerador de Software é um sistema meio, ou seja, é um sistema responsável por criar um sistema, portanto não possui controle sobre operações de períodos interativo ou não supervisionado.

Todo o processamento de dados é realizado de maneira assíncrona, ou seja, mesmo que duas operações sejam solicitadas ao mesmo tempo, o servidor será capaz de processá-las paralelamente.

O sistema ainda não possui nenhuma política de cópia de segurança e restauração de dados.

2.8 Requisitos de adaptação do local

Durante a inicialização do Gerador de Software o sistema operacional deve estar conectado à internet.

2.9 Funções do produto/sistema

- Criação de metaformularios(telas);
- Criação de metacampos (campos das telas);
- Cadastro de metadados (informações de metacampos);
- Listagem de metadados;
- Edição de metadados;
- Exclusão de metadados;

- Entrega contínua.

2.10 Características dos usuários

Os usuários chaves desse sistema são usuários com o dominio mínimo de lógica de programação e modelagem de dados, não exigindo nenhum outro conhecimento técnico para utilização do sistema, porém, como o sistema é um criador de sistemas, não se pode afirmar qual a proficiência técnica ou grau de formação para outros tipos de usuários.

3 Requisitos Específicos

Abaixo serão apresentados os requisitos de usuários, funcionais, qualidade, regras de domínio, descrição de dados do sistema, diagrama de caso de uso, casos de uso descritivos e diagrama de domínio.

3.1 Requisitos de Usuários

Quadro 1 - Requisitos de usuário

Id	Descrição	Fonte
RU032	Criação de um *framework* que possibilite a criação de um cadastro de forma simplificada para um usuario leigo.	Murilo Peixoto
RU033	Criação de um *framework* que possibilite a edição de um cadastro de forma simplificada para um usuario leigo	Murilo Peixoto
RU034	Criação de um *framework* que possibilite a exclusão de um cadastro de forma simplificada para um usuario leigo	Murilo Peixoto
RU035	Criação de um *framework* que possibilite a exibição de um cadastro de forma simplificada para um usuario leigo	Murilo Peixoto
RU036	O *framework* deve possibilitara a vinculação de um cadastro a outro.	Eugênio Júlio M. Candido Carvalho

RU037	O sistema deve exibir um *dashboard* mostrando a quantidade de cadastros realizados.	Murilo Peixoto
RU038	O *framework* deve realizar qualquer tipo de nova funcionalidade no formata CD	Murilo Peixoto
RU039	O sistema deve ser capaz de manter um usuario logado após ele realizar login.	Murilo Peixoto
RU040	O sistema deve ter dois níveis de usuário.	Eugênio Júlio M. Candido Carvalho
RU041	O *framework* deve funcionar *WEB*, *Desktop* para *Windows*, *Desktop* para *Linux* e mobile para *Android.*	Murilo Peixoto
RU042	O sistema deve ter opção de criar um texto de sobre e adicionar o mesmo e ser refletido em tempo real.	Murilo Peixoto

Fonte: Elaborado pelo autor.

3.2 Requisitos Funcionais

Quadro 2 - RF001: Menu para usuários nivel root

Identificador	**Nome**
RF001	Menu para usuários nivel root

Caso de Uso	**Autor**
CSU55, CSU65	Murilo Barros Peixoto

Descrição

O menu para usuários nivel root deve conter *DashBoard,* Configurações (Cadastros e Configuração geral), Cadastros e Relatórios.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 3 - RF002: Menu para usuários nivel usuário

Identificador	**Nome**
RF002	Menu para usuários nivel usuário

Caso de Uso	**Autor**
CSU65	Murilo Barros Peixoto

Descrição
O menu para usuários nivel usuário deve conter Home, Cadastros e Relatórios.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 4 - RF003: Tela dinâmica de acordo com as especificações do metaformulario e metacampo no formato linear.

Identificador	Nome
RF003	Tela dinâmica de acordo com as especificações do metaformulario e metacampo no formato linear.

Caso de Uso	Autor
CSU65	Murilo Barros Peixoto

Descrição
Exibição do rotulo do metaformulario, construção de uma tabela com as colunas na ordem informada no cadastro de metacampos com o título de cada coluna como o nome do metacampo.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 5 - RF004: Realizar login

Identificador	Nome
RF004	Realizar login

Caso de Uso	Autor
CSU65	Murilo Barros Peixoto

Descrição

Para realização do login a aplicação devera se comunicar com a API do sistema no *endpoint* '/api/Login' utilizando o verbo POST passando pelo *body* o usuario e a senha no formato json.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 6 - RF005: Manter usuário logado com cacheamento

Identificador	Nome
RF005	Manter usuario logado com cacheamento.

Caso de Uso	Autor
CSU65	Murilo Barros Peixoto

Descrição

Para manter o usuário logado após realizar o login o sistema deve salvar a chave de sessão retornada após o login bem sucedido em um *Cookie* chamado CookieUsuario junto com todas as informações retornadas assim realizando o cacheamento para ele.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 7 - RF006: Cacheamento dos dados da pessoa do usuário

Identificador	Nome
RF006	Cacheamento dos dados da pessoa do usuário.

Caso de Uso	Autor
CSU65	Murilo Barros Peixoto

Descrição

O cacheamento dos dados da pessoa do usuário a aplicação após realizar o login deve comunicar-se com a API pelo *endpoint* ‘api/Pessoa/{sessaoKey}/{IdUsuario}’ utilizando o verbo GET passando em sua rota sessaoKey como id da sessão do usuário e IdUsuario com o id do usuario que está logado e salvar o retorno no *Cookie* CookiePessoa.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 8 - RF007: Cacheamento dos metaformularios para consulta

Identificador	Nome
RF007	Cacheamento dos metaformularios para consulta.

Caso de Uso	Autor
CSU65	Murilo Barros Peixoto

Descrição

O cacheamento dos metaformularios será realizado por meio de *Cookie*, para realizar o mesmo a aplicação após realizar o login devera se comunicar com a API no *endpoint* ‘api/Metaformulario’ utilizando o verbo GET passando como *query* sessaoKey como o id de sessão, tamanhoDaPagina como 100 e número da página como 1 e em seguida salvar o retorno no *Cookie* CookieFormularios.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 9 - RF008: Cacheamento dos metaformularios para consulta

Identificador	**Nome**
RF008	Cacheamento das configurações dinâmicas

Caso de Uso	**Autor**
CSU65	Murilo Barros Peixoto

Descrição

O cacheamento das configurações dinâmicas será realizado através de *Cookie* para isso a aplicação devera se comunicar com a API no *endpoint* ‘api/MetaFormulario/ {sessaoKey/Id’ utilizando o verbo GET passando em sua rota sessaoKey com id da sessão e id como o id do metaformulario de configuração dinâmica.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 10 - RF009: Novo usuário

Identificador	Nome
RF009	Novo usuario

Caso de Uso	Autor
CSU70	Murilo Barros Peixoto

Descrição

Para criar um usuário a aplicação devera se comunicar com a API do sistema no *endpoint* '/api/Usuario/CadastreUsuario' utilizando o verbo http (*Hyper Text Transfer Protocol*) POST passando pelo *body* o usuario, senha e confirmação da senha no formato json.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 11 - RF010: Logout

Identificador	**Nome**
RF010	*Logout*

Caso de Uso	**Autor**
CSU64	Murilo Barros Peixoto

Descrição

Ao realizar logout a aplicação deve apagar todos os *Cookies* usados e informar a API em *background* que a sessão atual já não é válida utilizando o *endpoint* ‘api/Sessao/Delete’.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 12 - RF011: Construção do menu

Identificador	Nome
RF011	Construção do menu

Caso de Uso	Autor
CSU65	Murilo Barros Peixoto

Descrição

Para a construção do menu a aplicação devera consultar o CookieFormularios para pegar os nomes do metaformularios salvos em cache.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 13 - RF012: Listagem de metaformularios

Identificador	Nome
RF012	Listagem de metaformularios

Caso de Uso	Autor
CSU55	Murilo Barros Peixoto

Descrição

Para a listagem de metaformularios a aplicação deverá realizar se comunicar com a API no *endpoint* ‘api/MetaFormulario’ utilizando o verbo GET passando como *query* sessaoKey como o id de sessão, tamanhoDaPagina como 10 e número da página como 1 e a paginação deve ser realizada por padrão de 10 em dez elementos.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 14 - RF013: Criação de metaformulario

Identificador	Nome
RF013	Criação de metaformulario

Caso de Uso	Autor
CSU61	Murilo Barros Peixoto

Descrição

Para criar um metaformulario a aplicação devera se comunicar com a API no *endpoint* ‘api/MetaFormulario’ utilizando o verbo POST passando no *body* o nome do formulário, rótulo, tipo do formulário e o id de sessão.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 15 - RF014: Edição de metaformulario

Identificador	Nome
RF014	Edição de metaformulario

Caso de Uso	Autor
CSU57	Murilo Barros Peixoto

Descrição

Para editar um metaformulario a aplicação devera se comunicar com a API no *endpoint* ‘api/MetaFormulario’ utilizando o verbo PUT passando no *body* o nome do formulário, rótulo e o tipo do formulário, id do metaformulario e o id de sessão.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 16 - RF015: Inativação de metaformulario

Identificador	Nome
RF015	Inativação de um metaformulario

Caso de Uso	Autor
CSU56	Murilo Barros Peixoto

Descrição

Para inativar um metaformulario a aplicação devera se comunicar com a API no *endpoint* ‘api/MetaFormulario/Delete’ utilizando o verbo PUT passando no *body* o id do metaformulario e o id de sessão.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 17 - RF016: Ativação de um metaformulario

Identificador	**Nome**
RF016	Ativação de um metaformulario

Caso de Uso	**Autor**
CSU71	Murilo Barros Peixoto

Descrição
Para ativar um metaformulario a aplicação devera se comunicar com a API no *endpoint* ‘api/MetaFormulario/Habilitar utilizando o verbo PUT passando no *body* o id do metaformulario e o id de sessão.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 18 - RF017: Listagem de metacampos

Identificador	**Nome**
RF017	Listagem de metacampos

Caso de Uso	**Autor**
CSU60	Murilo Barros Peixoto

Descrição
Para a listagem dos metacampos a aplicação deverá se comunicar com a API no *endpoint* ‘api/MetaCampo’ utilizando o verbo GET passando como *query* sessaoKey como o id de sessão, tamanhoDaPagina como 10 e número da página como 1 e a paginação deve ser realizada por padrão de 10 em dez elementos.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 19 - RF018: Criação de metacampo

Identificador	Nome
RF018	Criação de metacampo

Caso de Uso	Autor
CSU58	Murilo Barros Peixoto

Descrição

Para criar um metacampo a aplicação devera se comunicar com a API no *endpoint* ‘api/ MetaCampo’ utilizando o verbo POST passando no *body* o nome do campo, tipo, descrição, se é obrigatório, se é PK, ordem, tamanho, id do metaformulario que ele pertence, se está ativo, e id de sessão do usuário.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 20 - RF019: Edição de metacampo

Identificador	**Nome**
RF019	Edição de metacampo

Caso de Uso	**Autor**
CSU59	Murilo Barros Peixoto

Descrição
Para editar um metacampo a aplicação devera se comunicar com a API no *endpoint* ‘api/MetaCampo’ utilizando o verbo PUT passando *body* o nome do campo, tipo, descrição, se é obrigatório, se é PK, ordem, tamanho, id do metaformulario que ele pertence, se está ativo, e id de sessão do usuário.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 21 - RF022: Geração de PDF

Identificador	**Nome**
RF022	Geração de PDF

Caso de Uso	**Autor**
	Murilo Barros Peixoto

Descrição
Para geração genérica de um pdf de uma tabela será utilizado recurso o framework *DataTable* da linguagem JS.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 22 - RF023: Geração de PWA para Windows

Identificador	**Nome**
RF023	Geração de PWA para *Windows*

Caso de Uso	**Autor**
CSU63	Murilo Barros Peixoto

Descrição

Para geração do PWA para *Windows* a aplicação deve expor um *serviceworker* para que o *browser* que o usuário esteja acessando possa realizar a importação da aplicação por meio do endpoint ‘/*serviceworker* ,expor o endpoint ‘/offline.html’ para que a aplicação possa rodar *offline* e o endpoint ‘/manifest.webmanifest’ para que o *Windows* possa ter acesso as descrições do PWA.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 23 - RF024: Geração de PWA para Linux

Identificador	**Nome**
RF024	Geração de PWA para *Linux*

Caso de Uso	**Autor**
CSU63	Murilo Barros Peixoto

Descrição
Para geração do PWA para *Linux* a aplicação deve expor um *serviceworker* para que o *browser* que o usuário esteja acessando possa realizar a importação da aplicação por meio do endpoint '/*serviceworker* e expor o endpoint '/offline.html' para que a aplicação possa rodar *offline* e o endpoint '/manifest.webmanifest' para que o *Linux* possa ter acesso as descrições do PWA.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 24 - RF025: Geração de PWA para Android

Identificador	**Nome**
RF025	Geração de PWA para *Android*

Caso de Uso	**Autor**
CSU63	Murilo Barros Peixoto

Descrição
Para geração do PWA para *Android* a aplicação deve expor um *serviceworker* para que o *browser* que o usuário esteja acessando possa realizar a importação da aplicação por meio do endpoint ‘/*serviceworker* e expor o endpoint ‘/offline.html’ para que a aplicação possa rodar *offline* e o endpoint ‘/manifest.webmanifest’ para que o *Android* possa ter acesso as descrições do PWA.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 25 - RF026: Geração de relatório geral.

Identificador	Nome
RF026	Geração do relatório geral.

Caso de Uso	Autor
CSU62	Murilo Barros Peixoto

Descrição

Para exibir os dados do relatório a aplicação devera se comunicar com a API por meio do endpoint ‘api/DashBoard/CountMetaForm{sessaoKey}/{IdMetaFormulario}’ utilizando o verbo GET informando como rota o id de sessão do usuario logado e o id do metaformulario.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 26 - RF027: Impressão de relatório geral

Identificador	Nome
RF027	Impressão do relatório geral.

Caso de Uso	Autor
CSU62	Murilo Barros Peixoto

Descrição
A impressão do relatório geral deve-se utilizar o mecanismo nativo de impressão do JS.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 27 - RF028: Geração do DashBoard

Identificador	Nome
RF028	Geração do *DashBoard*

Caso de Uso	Autor
CSU62	Murilo Barros Peixoto

Descrição

Para exibir os dados do *DashBoard* a aplicação devera se comunicar com a API por meio do endpoint ‘api/DashBoard/CountDB’ informando como rota o id de sessão do usuario logado.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 28 - RF029: Cadastro de uma tupla de um metaformulario

Identificador	**Nome**
RF029	Cadastro de uma tupla de um metaformulario

Caso de Uso	**Autor**
CSU66	Murilo Barros Peixoto

Descrição

Para realizar o cadastro de uma tupla de um metaformulario a aplicação devera se comunicar com a API pelo *endpoint* ‘api/MetaObjeto’ utilizando o verbo POST e passando em seu *body* um objeto com dois atributos, o primeiro atributo é chamado de ‘dados’ e nele é passado uma lista de objetos onde cada objeto é composto por id do metacampo, Ativo, id de sessão e informação. O segundo atributo é chamado de ‘sessaoKey’ e nesse parâmetro é passado o id da sessão do usuário.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 29 - RF030: Edição de uma tupla de um metaformulario

Identificador	**Nome**
RF030	Edição de uma tupla de um metaformulario

Caso de Uso	**Autor**
CSU67	Murilo Barros Peixoto

Descrição

Para realizar a edição de uma tupla de um metaformulario a aplicação devera se comunicar com a API pelo *endpoint* ‘api/MetaObjeto’ utilizando o verbo PUT e passando em seu *body* um objeto com dois atributos, o primeiro atributo é chamado de ‘dados’ e nele é passado uma lista de objetos onde cada objeto é composto por id do metacampo, Ativo, id do metadado, id de sessão, informação e o id da tupla. O segundo atributo é chamado de ‘sessaoKey’ e nesse parâmetro é passado o id da sessão do usuário.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 30 - RF031: Listagem das tuplas de um metaformulario

Identificador	**Nome**
RF031	Listagem das tuplas de um metaformulario

Caso de Uso	**Autor**
CSU69	Murilo Barros Peixoto

Descrição
Para realizar a listagem das tuplas de uma metaformulario o sistema deve se comunicar com a API pelo *endpoint* ‘api/MetaObjeto/Detalhe’ utilizando o verbo GET passando como *query* o id de sessão, o id do metaformulario, tamanho da página e o número da página.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 31 - RF032: Exclusão de uma tupla de um metaformulario

Identificador	**Nome**
RF032	Exclusão de uma tupla do metaformulario

Caso de Uso	**Autor**
CSU68	Murilo Barros Peixoto

Descrição

Para realizar a exclusão de uma tupla de uma metaformulario o sistema deve se comunicar com a API pelo *endpoint* ‘api/MetaObjeto/’ utilizando o verbo DELETE passado como rota o id da sessão e o id da tupla que será excluída para realização da exclusão.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

3.3 Requisitos de Qualidade

Quadro 32 - RQ043: Lei sobre tratamento de dados pessoais

Identificador	**Nome**
RQ043	Lei sobre tratamento de dados pessoais

Caso de Uso	**Autor**
CSU58, CSU59, CSU60, CSU61, CSU62, CSU64, CSU65, CSU66, CSU69	Murilo Barros Peixoto

Descrição

LEI Nº 13.709 Art. 1º Esta Lei dispõe sobre o tratamento de dados pessoais, inclusive nos meios digitais, por pessoa natural ou por pessoa jurídica de direito público ou privado, com o objetivo de proteger os direitos fundamentais de liberdade e de privacidade e o livre desenvolvimento da personalidade da pessoa natural.

Dependência	**Prioridade**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 33 - RQ044: Confidencialidade

Identificador	Nome
RQ044	Confidencialidade

Caso de Uso	Autor
CSU57, CSU58, CSU59, CSU60, CSU61, CSU62, CSU64, CSU65, CSU66, CSU69	Murilo Barros Peixoto

Descrição
As informações dos usuários são de total restrição a manipulação e visualização do próprio, e a mesma não pode ser usada para outro fim, apenas no sistema.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 34 - RQ045: Segurança de acesso

Identificador	Nome
RQ045	Segurança de acesso

Caso de Uso	Autor
CSU58, CSU59, CSU60, CSU65, CSU66, CSU69, CSU70	Murilo Barros Peixoto

Descrição

As senhas dos usuários são criptografadas a fim de assegurar suas informações pessoais, tanto como seus dados de sessão e seus acessos no sistema.

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

Quadro 35- RQ046: Usabilidade

Identificador	Nome
RQ046	Usabilidade

Caso de Uso	Autor
CSU55	Murilo Barros Peixoto

Descrição
As telas do sistema são responsivas e capaz de se adaptar a tamanhos relacionados a dispositivos mobile, dispositivos desktop e navegadores *Web.*

Dependência	Prioridade
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	Essencial

Fonte: Elaborado pelo autor.

3.4 Regras de Domínio

Quadro 36 - RD047: Metacampo principal de um metaformulario

Identificador	**Nome**
RD47	Metacampo principal de um metaformulario

Caso de Uso	**Autor**
CSU60, CSU70	Murilo Barros Peixoto

Descrição
O metacampo principal de um metaformulario é o campo que possui o menor valor do atributo ordem.

Dependência	**Fonte**
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	O autor

Fonte: Elaborado pelo autor.

Quadro 37 - RD048: Metadado principal de uma tupla

Identificador	Nome
RD48	Metadado principal de uma tupla

Caso de Uso	Autor
CSU66, CSU69	Murilo Barros Peixoto

Descrição
O metadado principal de uma tupla é o metadado pertencente ao metacampo principal do metaformulario que o metadado pertence

Dependência	Fonte
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	O autor

Fonte: Elaborado pelo autor.

Quadro 38 - RD049: Exibição do metadado principal de uma tupla

Identificador	Nome
RD49	Exibição do metadado principal de uma tupla

Caso de Uso	Autor
	Murilo Barros Peixoto

Descrição

Caso o metadado principal seja de um metacampo do tipo dependente, então deverá ser exibido o metadado principal referente a tupla referenciada como dependente aplicando-se essa regra recursivamente. Caso não, deverá ser exibido o próprio metadado.

Dependência	Fonte
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	O autor.

Fonte: Elaborado pelo autor.

Quadro 39 - RD050: Nome de metaformularios

Identificador	Nome
RD50	Nome de metaformularios

Caso de Uso	Autor
CSU57, CSU61	Murilo Barros Peixoto

Descrição

Ao realizar o cadastro ou edição de um nome de um metaformulario o sistema devera conferir se o nome informado é igual ao ‘NomeFormularioConfiguracaoDinamica’ presente na enumeração ‘NomesReservados’ pois nenhum metaformulario deve possuir esse nome.

Dependência	Fonte
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	O autor.

Fonte: Elaborado pelo autor.

Quadro 40 - RD051: Validação de CPF

Identificador	Nome
RD51	Validação de CPF

Caso de Uso	Autor
CSU66, CSU69	Murilo Barros Peixoto

Descrição

Ao cadastrar um metacampo do tipo CPF o sistema deve validar o mesmo de acordo com o algoritmo presente nesse link: http://www.macoratti.net/alg_cpf.htm

Dependência	Fonte
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	O autor.

Fonte: Elaborado pelo autor.

Quadro 41 - RD052: Validação de CNPJ

Identificador	Nome
RD52	Validação de CNPJ

Caso de Uso	Autor
CSU66, CSU69	Murilo Barros Peixoto

Descrição

Ao cadastrar um metacampo do tipo CNPJ o sistema deve validar o mesmo de acordo com o algoritmo presente nesse link: http://www.macoratti.net/alg_cnpj.htm

Dependência	Fonte
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	O autor.

Fonte: Elaborado pelo autor.

Quadro 42 - RD053: Validação de E-MAIL

Identificador	Nome
RD53	Validação de E-MAIL

Caso de Uso	Autor
CSU66, CSU69	Murilo Barros Peixoto

Descrição

Ao cadastrar um metacampo do tipo e-mail o sistema deve validar o mesmo de acordo com o algoritmo de validação de e-mail da biblioteca *mail* do *framework .Net Core.*

Dependência	Fonte
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	O autor.

Fonte: Elaborado pelo autor.

Quadro 43 - RD054: Listagem

Identificador	Nome
RD54	Listagem

Caso de Uso	Autor
CSU55, CSU69	Murilo Barros Peixoto

Descrição
Toda listagem que faz comunicação com a API desse ser feita de forma página começando sempre com o número de página 1 e com o tamanho de página 10;

Dependência	Fonte
Todos os requisitos aplicado ao(s) caso(s) de uso(s) do requisito atual.	O autor.

Fonte: Elaborado pelo autor.

3.5 Descrição dos Dados do Sistema

Quadro 44 - DD072: Metadado

Identificador	Descrição
DD072	Metadado

Identificador	Tamanho	Tipo	Formato	Descrição
Id		Guid		Identificador
Ativo		Booleano		Identifica se o campo está ativo ou não.
DataCriacao		Data		Data de criação
Informação	100	Texto		Informação do metadado relacionado ao metacampo do metaformulario
IdMetaCampo		Guid		Id do meta campo
TuplaId		Guid		Identificador da Tupla

Fonte: Elaborado pelo autor.

Quadro 45 - DD073: Metaformulario

Identificador	Descrição
DD073	Metaformulario

Identificador	Tamanho	Tipo	Formato	Descrição
Id		Guid		Identificador
Ativo		Booleano		Identifica se o campo está ativo ou não.
DataCriacao		Data		Data de criação
Nome		Texto		Descrição representada como única
Rotulo		Texto		Descrição para o sistema
TipoFormulario		Numérico		Tipo de metaformulario

Fonte: Elaborado pelo autor.

Quadro 46 - DD074: Pessoa

Identificador	Descrição
DD074	Pessoa

Identificador	Tamanho	Tipo	Formato	Descrição
Id		Guid		Identificador
Ativo		Booleano		Identifica se o campo está ativo ou não.
DataCriacao		Data		Data de criação
Nome		Texto		Nome da pessoa
TipoPessoa		Numérico		Tipo da pessoa

Fonte: Elaborado pelo autor.

Quadro 47 - DD075: Sessao

Identificador	Descrição
DD075	Sessao

Identificador	Tamanho	Tipo	Formato	Descrição
Id		Guid		Identificador
Ativo		Booleano		Identifica se o campo está ativo ou não.
DataCriacao		Data		Data de criação
IdUsuario		Guid		Identificador do usuário

Fonte: Elaborado pelo autor.

Quadro 48 - DD076: Usuario

Identificador	Descrição
DD076	Usuario

Identificador	Tamanho	Tipo	Formato	Descrição
Id		Guid		Identificador
Ativo		Booleano		Identifica se o campo está ativo ou não.
DataCriacao		Data		Data de criação
Login	255	Texto		Login do usuário
Senha	1000	Texto		Senha
IdPessoa		Guid		Identificador da pessoa
TipoUsuario		Numérico		Tipo de usuario

Fonte: Elaborado pelo autor.

Quadro 49 - DD077: Metacampo

Identificador	Descrição
DD077	MetaCampo

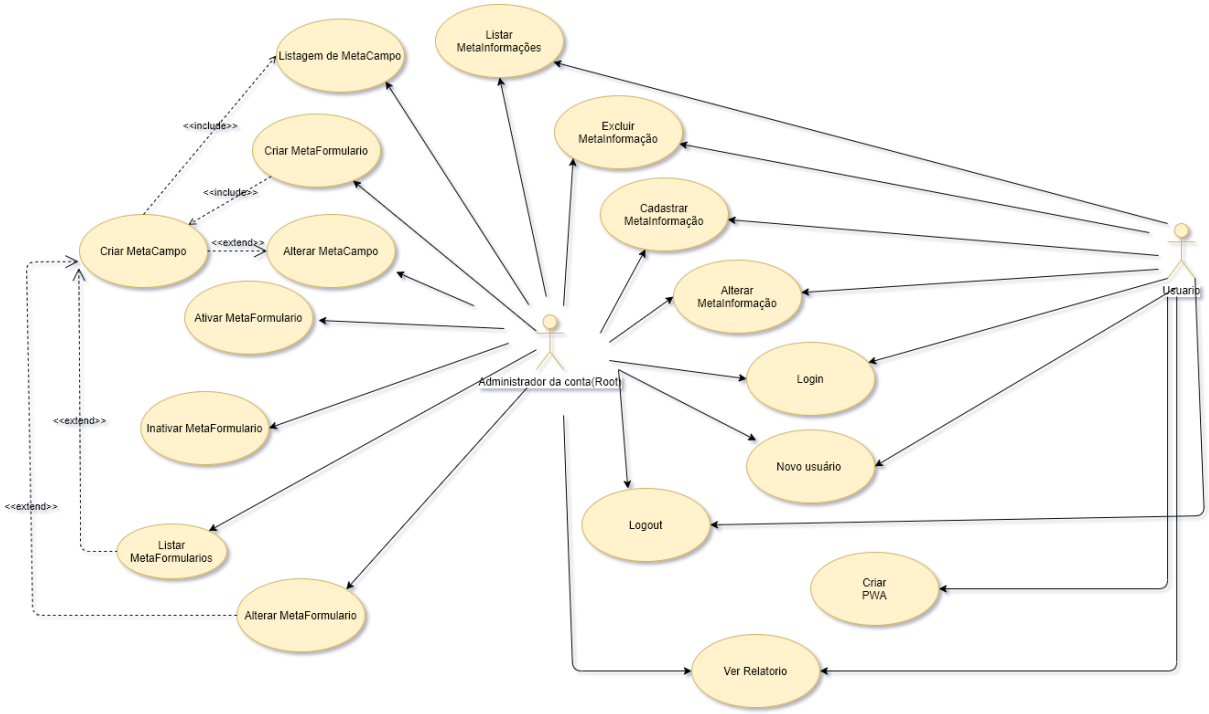
Identificador	Tamanho	Tipo	Formato	Descrição
Id		Guid		Identificador
Ativo		Booleano		Identifica se o campo está ativo ou não.
DataCriacao		Data		Data de criação
Nome	100	Texto		Nome do campo
Tipo		Numérico		Tipo do campo
Descrição	100	Texto		Descrição do campo
EhObrigatorio		Booleano		Identifica se o campo é obrigatório ou não
EhPk		Booleano		Identifica se o campo é identificador
Ordem		Booleano		Identifica a ordem do campo
Tamanho		Numérico		tamanho do campo.
IdMetaFormulario		Guid		Id do metaformulario que o meta campo pertence
IdMetaFormulario		Guid		Id metaformulario de chave

Fk				estrangeira

Fonte: Elaborado pelo autor.

3.6 Diagrama de Casos de Uso

Figura 2 - Diagrama de casos de uso



Fonte: Elaborado pelo autor.

3.7 Casos de Uso Descritivos

Identificador: CSU55

Nome: Listar Metaformularios

Requisito: RF012

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario root entra na tela de cadastro de metaformularios. O sistema comunica com a API para listar os metacampos cadastrados. O sistema processa o retorno da api e exibe os dados para o usuario.

Atores: Root

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Pós-condições:

A lista de metaformularios cadastrados em cache.

Cenário Principal -

1. O usuario abre a tela de cadastro de metaformulario.
2. O sistema comunica com a API pedindo os metaformularios cadastrados.
3. A API informa o sistema o resultado do pedido
4. O sistema valida o retorno da API.
5. O sistema processa a lista retornada da API.
6. O sistema exibe a lista de metaformularios para o usuario.

Cenários Alternativos -

3.a. O sistema não consegue uma comunicação com a API

3.a.1. O sistema informa ao usuario que não foi possível comunicar com a API.

4.a. O retorno da API é invalido.

4.a.1. O sistema informa ao usuario que o retorno da API é invalido.

5.a. O sistema termina o caso de uso pois a lista retornada é vazia.

Cenários de Exceção –

2.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: RQ046

Regras de Domínio: RD54

Identificador: CSU56

Nome: Inativar Metaformularios

Requisito: RF015

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario clica no botão inativar formulário para realizar a inativação dele. O sistema comunica com a API para executar essa ação. A API retorna informando que o metaformulario foi inativado e o sistema informa o usuario o resultado.

Atores: Root

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados dos metaformularios devem estar em cache.

Pós-condições:

O sistema atualiza os dados dos metaformularios presentes no cache.

Cenário Principal -

1. O usuario clica no botão de inativar o metaformulario.
2. O sistema mostra uma *modal* de confirmação para o usuario.
3. O usuário confirma a inativação.
4. O sistema comunica com a API solicitando a inativação do metaformulario.
5. A API informa o sistema o resultado do pedido.
6. O sistema processa o resultado.
7. O sistema informa ao usuario que inativou o metaformulario com sucesso.

Cenários Alternativos -

3.a. O usuario cancela a inativação

3.a.1 O sistema fecha a modal aberta

4.a. O sistema não consegue uma comunicação com a API

3.a.1. O sistema informa ao usuario que não foi possível comunicar com a API.

Cenários de Exceção –

4.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: N/A

Regras de Domínio: N/A

Identificador: CSU57

Nome: Alterar metaformulario

Requisito: RF014

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario clica no botão editar metaformulario para realizar a alteração. O sistema abre página de alteração de metaformulario. O usuario altera os campos possíveis no metaformulario. O sistema salva as alterações.

Atores: Root

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados do metaformulario deve estar em cache.

Pós-condições:

O sistema atualiza os dados dos metaformularios presentes no cache da página anterior.

Cenário Principal -

1. O usuario clica no botão editar metaformulario.
2. O sistema abre página de alteração de metaformulario.
3. O sistema busca os dados do metaformulario presente em cache.
4. O usuario altera o nome do metaformulario.
5. O usuario altera o rotulo do metaformulario.
6. O usuario clica em salvar.
7. O sistema comunica com a API solicitando a alteração do metaformulario.
8. A API informa o sistema o resultado do pedido.
9. O sistema processa o resultado
10. O sistema informa ao usuario que alterou o metaformulario com sucesso.
11. O sistema retorna ao caso de uso **CSU74.**

Cenários Alternativos -

6.a O usuario deixa o nome do metaformulario em branco.

6.a.1 O sistema informa ao usuario que o nome do metaformulario é obrigatório.

6.b O usuario deixa o rotulo do metaformulario em branco.

6.b.1 O sistema informa ao usuario que o rotulo do metaformulario é obrigatório.

6.c O usuario clica no botão voltar

6.c.1 O usuario clica no botão voltar.

6.c.1 O sistema redireciona o usuario para o caso de uso **CSU74.**

10.a O rotulo do metaformulario já existe.

10.a.1 O sistema informa o que o rotulo escolhido para o metaformulario já existe.

10.b O identificador do metaformulario não foi encontrado

10.b.1 O sistema informa ao usuário que o identificador informado não existe na base de dados.

10.c O usuario altera o tipo do metaformulario.

10.c.1 O sistema informa ao usuario que não é possível alterar o tipo do metaformulario.

Cenários de Exceção –

2.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: RQ044

Regras de Domínio: RD50

Identificador: CSU58

Nome: Criar Metacampo

Requisito: RF018

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario informa o nome do metacampo, o tipo como texto livre, a quantidade de caracteres, descrição, ordem, se é obrigatório, se é único e se é um campo dependente. O usuario clica em salvar. O sistema salva as informações.

Atores: Root

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados do metaformulario devem estar em cache.

Pós-condições:

O sistema atualiza os dados dos metaformularios presentes no cache da página anterior.

Cenário Principal -

1. O usuario informa o nome do metacampo.
2. O usuario informa o tipo como texto livre.
3. O usuario informa a quantidade de caracteres.
4. O usuario informa a descrição.
5. O usuario informa ordem.
6. O usuario informa se é obrigatório.
7. O usuario informa se é único.
8. O usuario informa que não é um campo dependente.
9. O usuário clica em salvar.
10. O sistema comunica com a API.
11. A API informa o sistema o resultado do pedido.
12. O sistema processa o resultado.
13. O sistema informa ao usuario que criou o metacampo com sucesso.
14. O sistema limpa os dados da página.
15. O sistema vai para o caso de uso **CSU60**

Cenários Alternativos –

2.a O usuario informa o tipo diferente de texto livre

2.a.1 O sistema inativa o campo quantidade de caracteres.

2.a.2 O sistema vai para o passo 4 do caso de uso.

8.a O usuario informa que é um metacampo dependente.

8.a.1 O sistema busca no cache os metaformularios cadastrados.

8.a.1.a O sistema não encontra os dados do metaformularios em cache.

8.a.1.a.1 O sistema busca na API os metaformularios cadastrados.

8.a.1.a.2 API informa o sistema o resultado do pedido.

8.a.1.a.3 O sistema processa o retorno da API.

8.a.1.a.3 O sistema retorna ao passo 8.a.1 do caso de uso.

8.a.2 O sistema lista os metaformularios cadastrados.

8.a.2.a A lista de metaformularios está vazia.

8.a.2.a.1 O sistema informa ao usuario que ainda não existe metaformularios cadastrados.

8.a.2.a.2 O sistema retorna ao passo 8 do caso de uso.

8.a.3 O usuario seleciona um metaformulario cadastrado.

8.a.3 O sistema retorna ao passo 9 do caso de uso.

9.a O usuario não informa o nome do metacampo

9.a.1 O sistema informa ao usuario que o nome é obrigatório.

9.a.2 O sistema retorna ao passo 1 do caso de uso.

9.b O usuario não informa a descrição do metacampo

9.b.1 O sistema informa ao usuario que a descrição é obrigatória.

9.b.2 O sistema retorna ao passo 4 do caso de uso.

9.c O usuario informa uma ordem existente.

9.c.1 O sistema informa ao usuario que a ordem já existe.

9.c.2 O sistema retorna ao passo 5 do caso de uso.

13.a O sistema informa o usuário que não foi possível criar o metacampo.

13.a.1 O sistema retorna ao passo 1 do caso de uso.

13.b O nome do metacampo já existe

13.b.1 O sistema informa o usuário o nome do metacampo já existe.

13.b.2 O sistema retorna ao passo 1 do caso de uso.

Cenários de Exceção –

8.a.1.a.1 Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: RQ043, RQ044, RQ045

Regras de Domínio: N/A

Identificador: CSU59

Nome: Alterar MetaCampo

Requisito: RF019

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario altera o nome do metacampo, o tipo como texto livre, a quantidade de caracteres, descrição, ordem, se é obrigatório, se é único e se é um campo dependente. O usuario clica em salvar. O sistema salva as informações.

Atores: Root

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados do metaformulario devem estar em cache.

Pós-condições:

O sistema atualiza os dados dos metaformularios presentes no cache da página anterior.

Cenário Principal -

1. O usuario altera o nome do metacampo.
2. O usuario informa o tipo como texto livre.
3. O usuario informa a quantidade de caracteres.
4. O usuario informa a descrição.
5. O usuario informa ordem.
6. O usuario informa se é obrigatório.
7. O usuario informa se é único.
8. O usuario informa que não é um campo dependente.
9. O usuário clica em salvar.
10. O sistema comunica com a API.
11. A API informa o sistema o resultado do pedido.
12. O sistema processa o resultado.
13. O sistema informa ao usuario que criou o metacampo com sucesso.
14. O sistema limpa os dados da página.
15. O sistema vai para o caso de uso **CSU60**.

Cenários Alternativos –

2.a O usuario informa o tipo diferente de texto livre

2.a.1 O sistema inativa o campo quantidade de caracteres.

2.a.2 O sistema vai para o passo 4 do caso de uso.

8.a O usuario informa que é um metacampo dependente.

8.a.1 O sistema busca no cache os metaformularios cadastrados.

8.a.1.a O sistema não encontra os dados do metaformularios em cache.

8.a.1.a.1 O sistema busca na API os metaformularios cadastrados.

8.a.1.a.2 API informa o sistema o resultado do pedido.

8.a.1.a.3 O sistema processa o retorno da API.

8.a.1.a.3 O sistema retorna ao passo 8.a.1 do caso de uso.

8.a.2 O sistema lista os metaformularios cadastrados.

8.a.2.a A lista de metaformularios está vazia.

8.a.2.a.1 O sistema informa ao usuario que ainda não existe metaformularios cadastrados.

8.a.2.a.2 O sistema retorna ao passo 8 do caso de uso.

8.a.3 O usuario seleciona um metaformulario cadastrado.

8.a.3 O sistema retorna ao passo 9 do caso de uso.

9.a O usuario informa uma ordem que já existe

9.a.1 O sistema informa ao usuario que a ordem já existe.

9.a.2 O sistema retorna ao passo 4 do caso de uso.

13.a O sistema informa o usuário que não foi possível criar o metacampo.

13.a.1 O sistema retorna ao passo 1 do caso de uso.

13.b O sistema informa o usuário o nome do metacampo já existe

13.b.1 O sistema retorna ao passo 1 do caso de uso.

13.c O sistema informa o usuário que o metacampo não pode alterar o tipo pois ele já possui metadados vinculados a ele.

13.c.1 O sistema retorna ao passo 1 do caso de uso.

Cenários de Exceção –

8.a.1.a.1 Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: RQ043, RQ044, RQ045

Regras de Domínio: N/A

Identificador: CSU60

Nome: Listagem de metacampos

Requisito: RF017

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario root entra na tela de cadastro de metacampos. O sistema comunica com a API para listar os metacampos cadastrados. O sistema processa o retorno da api e exibe os dados para o usuario

Atores: Root

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

O dado do metaformulario relacionado ao metacampo deve estar em cache.

Pós-condições:

A lista de metacampos cadastrados em cache.

Cenário Principal -

1. O usuario abre a tela de cadastro de metacampos.
2. O sistema comunica com a API pedindo os metacampos cadastrados para o metaformulario informado.
3. A API informa o sistema o resultado do pedido
4. O sistema valida o retorno da API.
5. O sistema processa a lista retornada da API.
6. O sistema exibe a lista de metaformularios para o usuario.

Cenários Alternativos -

3.a. O sistema não consegue uma comunicação com a API

3.a.1. O sistema informa ao usuario que não foi possível comunicar com a API.

4.a. O retorno da API é invalido.

4.a.1. O sistema informa ao usuario que o retorno da API é invalido.

5.a. O sistema termina o caso de uso pois a lista retornada é vazia.

Cenários de Exceção –

2.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: RQ043, RQ044, RQ045

Regras de Domínio: RD47

Identificador: CSU61

Nome: Criar Metaformularios

Requisito: RF013

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuário informa o nome do metaformulario, o rotulo e o tipo (Linear ou Multidados). O usuario clica em salvar. O sistema informa o usuario que o metaformulario foi salvo com sucesso.

Atores: Root

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados dos metaformularios devem estar em cache.

Pós-condições:

O sistema atualiza os dados dos metaformularios presentes no cache.

Cenário Principal -

1. O usuário informa o nome do metaformulario.
2. O usuário informa o rotulo do metaformulario.
3. O usuário informa o tipo do metaformulario.
4. O usuário clica em salvar.
5. O sistema comunica com a API informando os dados do metaformulario.
6. A API informa o sistema o resultado do pedido
7. O sistema valida o retorno da API.
8. O sistema informa o usuário que o metaformulario foi salvo.

Cenários Alternativos –

5.a. O sistema não consegue uma comunicação com a API

5.a.1. O sistema informa ao usuario que não foi possível comunicar

com a API.

6.a. O retorno da API é invalido.

6.a.1. O sistema informa ao usuario que o retorno da API é invalido.

8.a O nome do metaformulario já existe

8.a.1 O sistema informa o usuario que o nome do metaformulario já existe.

8.a.2 O sistema retorna ao passo 1 do caso de uso.

Cenários de Exceção –

5.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: RQ043, RQ044

Regras de Domínio: RD50

Identificador: CSU62

Nome: Ver relatório

Requisito: RF026, RF027, RF028

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuário seleciona o menu a opção de ver o relatório. O sistema busca as informações do relatório. O sistema exibe o relatório para o usuário.

Atores: Root, Usuário

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Pós-condições:

Os dados do relatório estarão cacheados no sistema.

Cenário Principal -

1. O usuário seleciona o menu a opção de ver o relatório.
2. O sistema comunica com a API para obter o relatório.
3. A API informa o sistema o resultado do pedido
4. O sistema valida o retorno da API.
5. O sistema exibe o relatório para o usuário.

Cenários Alternativos –

N/A

Cenários de Exceção –

2.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: RQ043, RQ044

Regras de Domínio: N/A

Identificador: CSU63

Nome: Criar PWA

Requisito: RF023, RF024, RF025

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuário seleciona a opção de baixar/instalar o software em seu dispositivo (*Desktop Windows, Desktop Linux ou Android*). O sistema comunica com o sistema operacional da máquina do usuario. O sistema operacional instala do PWA.

Atores: Root, Usuário

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Pós-condições:

PWA instalado no dispositivo do usuário.

Cenário Principal -

1. O usuário seleciona a opção de baixar/instalar o software em seu dispositivo.
2. O sistema comunica com o sistema operacional da máquina do usuario.
3. O sistema operacional instala do PWA.

Cenários Alternativos –

2.a O sistema não consegue comunicação com o sistema operacional.

2.a.1 O sistema informa ao usuario que não conseguiu comunicação com o dispositivo dele.

2.b O dispositivo do usuário não suporta essa funcionalidade.

2.b.1 O sistema informa ao usuario seu dispositivo não suporta essa funcionalidade.

Cenários de Exceção –

3.a O sistema operacional do usuario não é capaz de instalar o PWA.

Requisitos Relacionados: N/A

Regras de Domínio: N/A

Identificador: CSU64

Nome: Logout

Requisito: RF010

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario ou sistema informa que deseja realizar logout. O sistema apaga todos os dados de sessão do usuário. O sistema redireciona o usuario para a tela de login.

Atores: Root, Usuário

Pré-condições:

A sessão do usuário deve estar ativa.

Pós-condições:

Dados da sessão do usuário apagados

Cenário Principal -

1. O usuario ou sistema informa que deseja realizar logout.
2. O sistema comunica com a API a realização do logout.
3. O sistema apaga os todos *Cookies* relacionados a sessão.
4. O sistema redireciona o usuario para a tela de login.

Cenários Alternativos –

N/A

Cenários de Exceção –

N/A

Requisitos Relacionados: RQ043, RQ044

Regras de Domínio: N/A

Identificador: CSU65

Nome: Login

Requisito: RF001, RF002, RF003, RF004, RF005, RF006, RF007, RF008, RF011

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario informa seu login e senha e clica em logar.
O sistema valida as informações. O sistema informa o usuario que o login foi realizado com sucesso e realiza o cacheamento dos dados necessários.

Atores: Root, Usuário

Pré-condições:

Todos os *cookies* relacionados a sessão não devem existir.

Pós-condições:

Todos os *cookies* relacionados a sessão devem existir.

Cenário Principal –

1. O usuário informa seu login.
2. O usuário informa sua senha.
3. O usuário clica em login.
4. O sistema comunica com a API para validar os dados
5. A API informa o sistema o resultado do pedido.
6. O sistema valida o retorno da API.
7. O sistema informa o usuário que o login foi realizado com sucesso.
8. O sistema salva os dados retornados do login em cache utilizando *cookie*.
9. O sistema solicita a API os metaformularios.
10. O sistema valida o retorno da API.
11. O sistema salva em cache utilizando *cookie* os metaformularios.
12. O sistema verifica o tipo de usuário utilizando o *cookie* do retorno do login.
13. O sistema constrói o menu de acordo com o tipo de usuário.

Cenários Alternativos –

3.a O usuario não informa login ou senha

3.a.1 O sistema informa ao usuario que login e senha é obrigatório.

3.a.2 O sistema retorna ao passo 1 do caso de uso.

4.a. O sistema não consegue uma comunicação com a API

4.a.1. O sistema informa ao usuario que não foi possível comunicar com a API.

6.a O retorno da API é invalido.

6.a.1. O sistema informa ao usuario que o retorno da API é invalido.

7.a Usuário ou senha inválidos.

7.a.1 O sistema informa ao usuário que o login ou a senha dele não foi encontrado na base de dados.

7.a.2 O sistema retorna ao passo 1 do caso de uso.

10.a O retorno da API é invalido.

10.a.1 O sistema informa ao usuario que o retorno da API é invalido.

10.a.2 O sistema apaga os dados de cache salvos no passo 8.

10.a.3 O sistema retorna ao passo 1 do caso de uso.

Cenários de Exceção –

12.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: RQ043, RQ044, RQ045

Regras de Domínio: N/A

Identificador: CSU66

Nome: Cadastrar metainformação

Requisito: RF029

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario clica em cadastrar novo metadado. O sistema abre uma modal com os metacampos possíveis para o cadastro. O usuario informa os dados que deseja inserir em cada metacampo. O usuario clica em salvar metacampos. O sistema valida os metacampos e salva os dados.

Atores: root e usuário.

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados dos metaformularios devem estar em cache.

Pós-condições:

Lista de metainformações atualizada em cache

Cenário Principal –

1. O usuario clica em cadastrar novo metadado.
2. O sistema abre uma modal com todos os metadados do metaformulario.
3. O usuario informa os metadados relacionados a cada metacampo.
4. O usuario clica no botão salvar.
5. O sistema comunica com a API.
6. A API informa o sistema o resultado do pedido.
7. O sistema processa o resultado.
8. O sistema informa ao usuario que salvou os metadados.
9. O sistema retorna ao caso de uso **CSU69**

Cenários Alternativos –

2.a Não existe metadados cadastrados para o metaformulario.

2.a.1 O sistema informa ao usuário que não existe metacampos cadastrados para o metaformulario.

2.a.2 O sistema executa o passo 9 do caso de uso.

4.a O usuario não informou um campo obrigatório.

4.a.1 O sistema informa ao usuario que nem todos os campos obrigatórios foram preenchidos.

Cenários de Exceção –

5.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

5.b Se a API estiver fora do ar o sistema informa ao usuário que a houve um problema ao estabelecer a comunicação com a API.

Requisitos Relacionados: RQ043, RQ044, RQ045

Regras de Domínio: RD48, RD51, RD52, RD53

Identificador: CSU67

Nome: Alterar metainformação

Requisito: RF030

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario clica em alterar metainformação. O sistema abre uma modal com todos os dados relacionados a tupla selecionada. O usuario altera a metainformação e clica em salvar. O sistema salva os dados alterados.

Atores: root e usuário.

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados dos metaformularios devem estar em cache.

Pós-condições:

Lista de metainformações atualizada em cache

Cenário Principal –

1. O usuario clica em alterar metadado.
2. O sistema abre uma modal com todos os metadados do metaformulario já com seus metadados carregados de acordo com a sua tupla.
3. O usuario informa os metadados que deseja alterar de cada metacampo.
4. O usuario clica no botão salvar.
5. O sistema comunica com a API.
6. A API informa o sistema o resultado do pedido.
7. O sistema processa o resultado.
8. O sistema informa ao usuario que alterou os metadados.
9. O sistema retorna ao caso de uso **CSU69**

Cenários Alternativos –

4.a O usuario não informou um campo obrigatório.

4.a.1 O sistema informa ao usuario que nem todos os campos obrigatórios foram preenchidos.

Cenários de Exceção –

5.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

5.b Se a API estiver fora do ar o sistema informa ao usuário que a houve um problema ao estabelecer a comunicação com a API.

Requisitos Relacionados: RQ043, RQ044, RQ045

Regras de Domínio: RD48, RD51, RD52, RD53

Identificador: CSU68

Nome: Excluir metainformação

Requisito: RF032

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario clica em excluir metainformação. O sistema abre uma modal confirmando a exclusão. O usuário confirma a exclusão. O sistema exclui o metacampo.

Atores: root e usuário.

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados dos metaformularios devem estar em cache.

Pós-condições:

Lista de metainformação atualizada em cache

Cenário Principal –

1. O usuario clica em excluir metadado.
2. O sistema abre uma modal de confirmação ao usuário.
3. O usuario confirma a exclusão.
4. O sistema comunica com a API.
5. A API informa o sistema o resultado do pedido.
6. O sistema processa o resultado.
7. O sistema informa ao usuario que exclui o metadados.
8. O sistema retorna ao caso de uso **CSU69**

Cenários Alternativos –

3.a O usuario não confirma a exclusão.

3.a.1 O sistema fecha a modal e encerra o caso de uso.

Cenários de Exceção –

5.a Se o sistema não encontrar os dados de sessão do usuario o mesmo

vai para o caso de uso **CSU64**.

5.b Se a API estiver fora do ar o sistema informa ao usuário que a houve um problema ao estabelecer a comunicação com a API.

Requisitos Relacionados: N/A

Regras de Domínio: N/A

Identificador: CSU69

Nome: Listar metainformações

Requisito: RF031

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario abre a tela de um metaformulario. O sistema busca os metacampos e metadados relacionados ao mesmo. O sistema lista todos os metacampos de acordo com seus metacampos.

Atores: root e usuário.

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados dos metaformularios devem estar em cache.

Pós-condições:

Lista de metainformações carregada em cache

Cenário Principal –

1. O usuario abre a tela de um metaformulario.
2. O sistema comunica com a API pedindo os dados do metaformulario.
3. A API informa o sistema o resultado do pedido.
4. O sistema processa o resultado.
5. O sistema constrói uma tabela de acordo com os metacampos.
6. O sistema insere na tabela os metadados dos metacampos.

Cenários Alternativos –

6.a Não existe metadados cadastrados

6.a.1 O sistema encerra o caso de uso.

Cenários de Exceção –

5.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

5.b Se a API estiver fora do ar o sistema informa ao usuário que a houve

um problema ao estabelecer a comunicação com a API.

Requisitos Relacionados: RQ043, RQ044, RQ045

Regras de Domínio: RD48, RD51, RD52, RD53, RD54

Identificador: CSU70

Nome: Novo usuário

Requisito: RF009

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuário informa seu e-mail e senha. O sistema valida os dados. O sistema cria o usuário.

Atores: root e usuário.

Pré-condições:

N/A

Pós-condições:

Cenário Principal –

1. O usuario informa o email e a senha
2. O usuario clica em salvar.
3. O sistema comunica com a API informando o email e a senha.
4. A API informa o sistema o resultado do pedido.
5. O sistema processa o resultado.
6. O sistema informa ao usuário que criou sua conta.
7. O sistema redireciona o usuario para o caso de uso CSU65

Cenários Alternativos –

3.a Email já existe

3.a.1 O sistema informa que o email já existe

3.a.2 O sistema retorna ao passo 1

Cenários de Exceção –

4.a Se a API estiver fora do ar o sistema informa ao usuário que a houve um problema ao estabelecer a comunicação com a API.

Requisitos Relacionados: RQ045

Regras de Domínio: RD53

Identificador: CSU71

Nome: Ativar Metaformularios

Requisito: RF016

Responsável: Murilo Barros Peixoto

Descrição/Resumo: O usuario clica no botão ativar formulário para realizar a ativação dele. O sistema comunica com a API para executar essa ação. A API retorna informando que o metaformulario foi ativado e o sistema informa o usuario o resultado.

Atores: Root

Pré-condições:

O usuario deve estar previamente logado.

A sessão do usuario deve estar ativa.

Os dados dos metaformularios devem estar em cache.

Pós-condições:

O sistema atualiza os dados dos metaformularios presentes no cache.

Cenário Principal -

1. O usuario clica no botão de ativar o metaformulario.
2. O sistema mostra uma *modal* de confirmação para o usuario.
3. O usuário confirma a ativação.
4. O sistema comunica com a API solicitando a ativação do metaformulario.
5. A API informa o sistema o resultado do pedido.
6. O sistema processa o resultado.
7. O sistema informa ao usuario que ativou o metaformulario com sucesso.

Cenários Alternativos -

3.a. O usuario cancela a ativação

3.a.1 O sistema fecha a modal aberta

4.a. O sistema não consegue uma comunicação com a API

3.a.1. O sistema informa ao usuario que não foi possível comunicar com a API.

Cenários de Exceção –

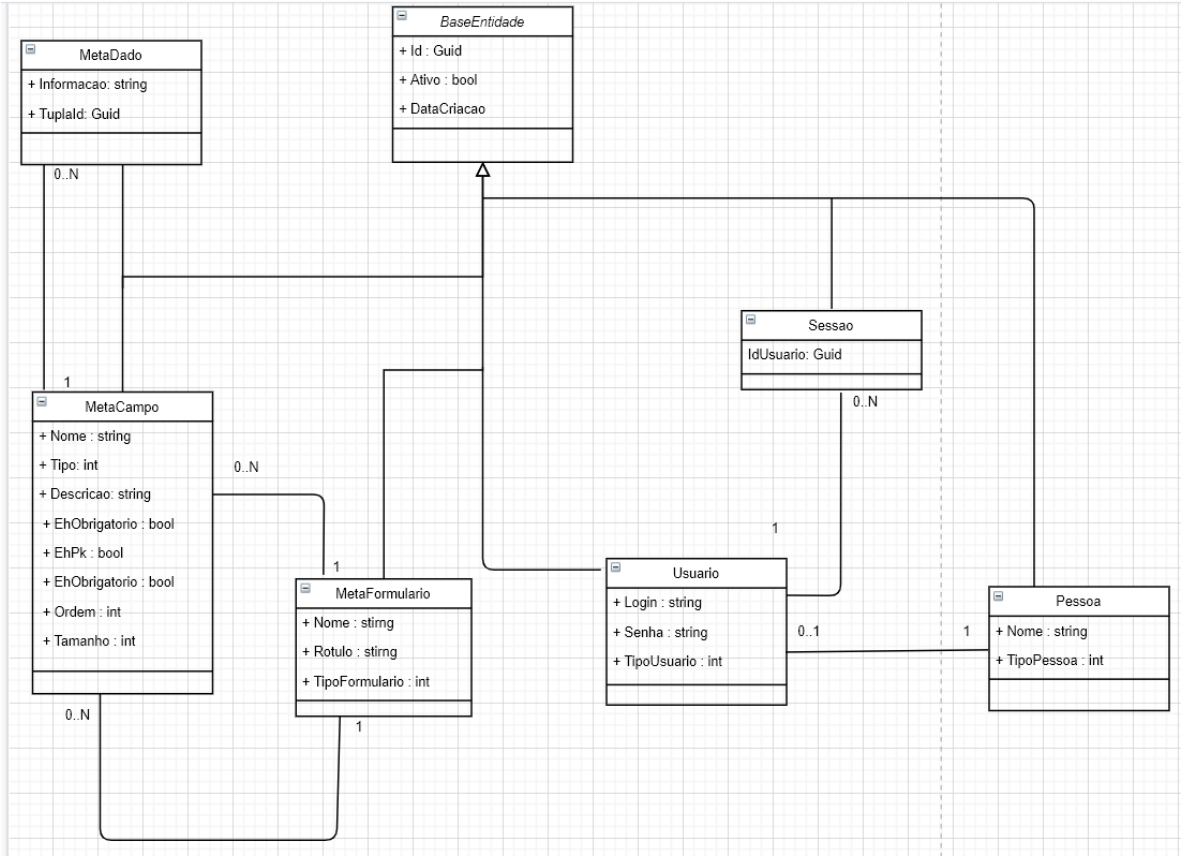
4.a Se o sistema não encontrar os dados de sessão do usuario o mesmo vai para o caso de uso **CSU64**.

Requisitos Relacionados: N/A

Regras de Domínio: N/A

3.8 Diagrama de Domínio

Figura 3 - Diagrama de domínio



Fonte: Elaborado pelo autor.

4 Desenvolvimento do sistema

Abaixo é apresentado as ferramentas e tecnologias utilizadas no desenvolvimento do sistema e os resultados obtidos.

4.1 Ferramentas utilizadas

As próximas subseções apresentam as ferramentas e tecnologias utilizadas para o desenvolvimento do sistema.

4.2 .NET Core

Todo material utilizado para definir .Net Core foi é segundo a Microsoft (2020)

.NET Core é um *framework* desenvolvido pela empresa Microsoft com o intuito de facilitar o desenvolvimento de softwares seguros e escaláveis e que podem ser executados em ambientes *Linux, Windows* e Mac*.* A escolha dessa ferramenta para o desenvolvimento se deu pelo fato de ela possuir recursos de baixa complexidade para o desenvolvimento, acesso a banco de dados e baixa curva de aprendizado devido ao fato da vasta quantidade de documentação oferecida pela empresa Microsoft.

De acordo com a documentação da Microsoft Com o .NET, seus arquivos de código e de projeto parecem e sentem o mesmo, independentemente do tipo de aplicativo que você está criando. Você tem acesso aos mesmos recursos de tempo de execução, API e linguagem com cada aplicativo.

De acordo O acesso da dados podem ser feito utilizando a biblioteca LINQ (*Language Integrated Query*) do .NET Core que permite criar consultas que independem do banco de dados utilizado facilitando assim o baixo acoplamento da aplicação em relação ao seu banco de dados.

A consulta integrada à linguagem (LINQ) permite que você escreva um código declarativo para operar em dados. Os dados podem ser de várias formas (como objetos na memória, um banco de dados SQL ou um documento XML), mas o código LINQ que você escreve não difere para cada fonte de dados.

4.3 C#

De acordo com (Microsoft,2020) C# é uma linguagem de programação moderna, orientada a objeto e de tipo seguro. O C# tem suas raízes na família de linguagens C e os programadores em C, C++, Java e Javascript a reconhecerão imediatamente.

C# é uma linguagem orientada a objetos porem ela pode assimilar a uma linguagem multi-paradigma, pois a mesma embora seja fortemente tipada tem a possibilidade de se trabalhar com tipos não definidos além do fato que ela possui a possibilidade de trabalhar com tipos de inferência.

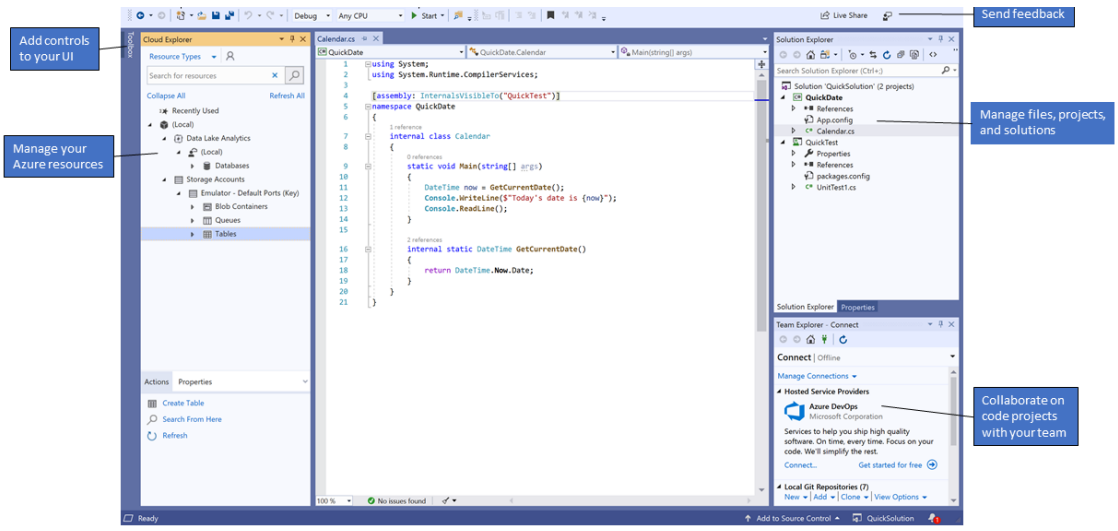
Os programas em C# podem ser armazenados em vários arquivos de origem. Quando um programa em C# é compilado, todos os arquivos de origem são processados em conjunto e os arquivos de origem podem fazer referência livremente entre si. Conceitualmente, é como se todos os arquivos de origem fossem concatenados em um arquivo grande antes de serem processados. Declarações de encaminhamento nunca são necessárias em C# porque, com poucas exceções, a ordem de declaração é insignificante. O C# não limita um arquivo de origem para declarar apenas um tipo público, nem exige o nome do arquivo de origem para corresponder a um tipo declarado no arquivo de origem (Microsoft,2020).

4.4 *Visual Studio*

O ambiente de desenvolvimento integrado do *Visual Studio* é um painel de inicialização criativo que você pode usar para editar, depurar e compilar o código e, em seguida, publicar um aplicativo. Um IDE (ambiente de desenvolvimento integrado) é um programa repleto de recursos que pode ser usado por muitos aspectos do desenvolvimento de software. Além do editor e do depurador padrão fornecidos pela maioria dos IDEs, o Visual Studio inclui compiladores, ferramentas de preenchimento de código, designers gráficos e muitos outros recursos para facilitar o processo de desenvolvimento de *software* (Microsoft, 2020).

A figura 20 representa como é o ambiente de desenvolvimento utilizado para a criação do projeto.

Figura 4 - Imagem da IDE



Fonte: (Microsoft,2020)

4.5 SQL Server

O banco de dados utilizado foi o SQL Server da própria Microsoft, pois, como todo o ferramental enunciado acima pertence a ela a comunicação entre a aplicação e o banco de dados se faz de maneira natural.

Para a manipulação do banco de dados foi utilizado o a IDE do SSMS (*SQL Server Management Studio*) da própria Microsoft. O SSMS é um ambiente integrado para gerenciar qualquer infraestrutura de SQL, do SQL Server para o Banco de Dados SQL do Azure. O SSMS fornece ferramentas para configurar, monitorar e administrar instâncias do SQL Server e bancos de dados (Microsoft,2020).

4.6 EFC *(Entity Framework Core)*

Para a manipulação do banco de dados por parte da aplicação foi utilizado o *framework* EFC, tanto para criação de esquemas de banco, tabelas, colunas, alterações de campos e manipulação de dados como *insert*, *update*, *delete* e *select.* O EFC usa como linguagem de escrita de consultas SQL a biblioteca LINQ do próprio .NET Core, porém, caso necessário do uso de uma consulta no formato SQL o EFC possui em seu núcleo suporte para leitura e processamento de SQL.

O Entity Framework Core é um mapeador moderno de banco de dados de objeto para .NET. Ele dá suporte a consultas LINQ, controle de alterações, atualizações e migrações de esquema (Microsoft,2020).

O Entity Framework usa um conjunto de convenções para criar um modelo com base na forma de suas classes de entidade. Você pode especificar configurações adicionais para complementar e/ou substituir o que foi descoberto por convenção (Microsoft,2020).

Para o desenvolvimento ao lado do EFC foi utilizado o paradigma *Code First,* ele trabalha com a criação das classes antes do banco de dados e o EFC interpreta essa classe para modelar e criar o modelo banco. O *Code First* é comumente usado quando se quer ter um controle maior há nível de código-fonte do modelo de dados gerado (Alves,2013*).*

O EFC utiliza atributos conhecidos na linguagem C# como Anotações para definir como uma propriedade de uma classe será construída no banco de dados como por exemplo criação de uma chave primaria de uma tabela, um propriedade obrigatório chave uma estrangeira etc. Você também pode aplicar atributos (conhecidos como Anotações de Dados) a classes e propriedades (Microsoft,2020).

4.7 Asp.NET Core MVC

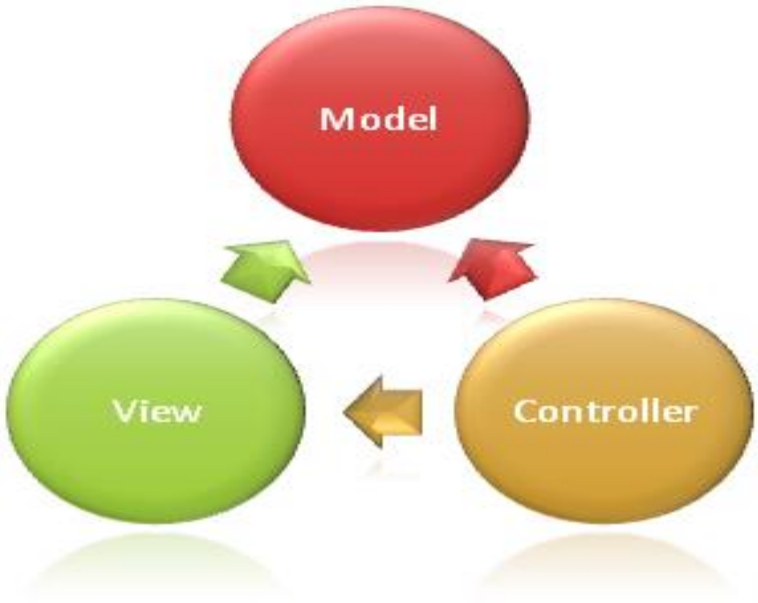
Todo material utilizado para definir Asp.Net MVC Core foi é segundo a Microsoft (2020)

O ASP.NET Core MVC é uma estrutura avançada para a criação de aplicativos Web e APIs usando o padrão de design *Model-View-Controller*.

O padrão de arquitetura MVC (Model-View-Controller) separa um aplicativo em três grupos de componentes principais: Modelos, Exibições e Componentes. Esse padrão ajuda a obter a separação de interesses. Usando esse padrão, as solicitações de usuário são encaminhadas para um Controlador, que é responsável por trabalhar com o Modelo para executar as ações do usuário e/ou recuperar os resultados de consultas. O Controlador escolhe a Exibição a ser exibida para o usuário e fornece-a com os dados do Modelo solicitados.

A figura 21 mostra como o padrão MVC se divide e como é a comunicação é realizada entre cada camada.

Figura 5 - MVC



Fonte: (Microsoft,2020)

A camada de *Model* tem a responsabilidade de realizar a lógica de negócios ou qualquer operação que seja necessário a persistência de dados.

A camada de *View* possui a responsabilidade de apresentação, ou seja, ela apresenta as interfaces gráficas ao usuário.

A camada de *Controller* é a camada que realiza a comunicação entre o *Model* e a *View,* porém, a *View* pode instanciar diretamente a *Model* para realizar operações diretas sem a necessidade de aumentar a comunicação com o *Controller.*

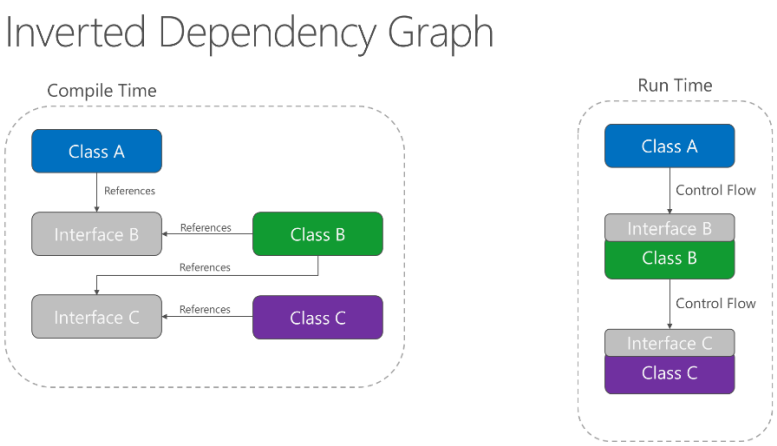
O ASP.NET Core MVC fornece uma maneira com base em padrões para criar sites dinâmicos que habilitam uma separação limpa de preocupações.

O ASP .NET Core MVC traz consigo o padrão de Injeção de dependências, o mesmo pode ser utilizado para trabalhar com o padrão IoC (Inversão de controle) esse padrão tem por finalidade tirar a responsabilidade de negócio da aplicação e entregar a outro modulo do sistema para realizar o gerenciamento dele. A inversão de dependência é uma parte fundamental da criação de aplicativos menos rígidos, já que os detalhes da implementação podem ser escritos para depender e implementar abstrações de nível superior, em vez do contrário. Como resultado, os aplicativos resultantes são mais testáveis, modulares e de manutenção mais fácil. A prática da injeção de dependência se tornou possível pela observância do princípio da inversão de dependência.

A aplicação do princípio da inversão de dependência permite que A chame métodos em uma abstração implementada por B, possibilitando que A chame B em runtime, mas que B dependa de uma interface controlada por A em tempo de compilação (invertendo, portanto, a dependência típica em tempo de compilação). Em tempo de execução, o fluxo da execução do programa permanece inalterado, mas a introdução de interfaces significa que diferentes implementações dessas interfaces podem ser conectadas com facilidade.

A figura 22 mostra a diferença do comportamento com o padrão injeção de dependência e sem ele.

Figura 6 - Inversão de dependência



Fonte: (Microsoft,2020)

O Asp.Net Core faz uso de *Cookie* para o controle de sessão de usuários. Depois que um cookie é criado, o cookie é a única fonte de identidade. Se uma conta de usuário estiver desabilitada em sistemas o sistema de cookie autenticação do aplicativo continua processando solicitações com base na autenticação cookie e O usuário permanece conectado ao aplicativo, contanto que a autenticação cookie seja válida. Uma abordagem para a cookie validação é baseada na manutenção do controle quando o banco de dados do usuário é alterado. Se o banco de dados não tiver sido alterado desde que o usuário cookie foi emitido, não haverá necessidade de autenticar novamente o usuário se ele cookie ainda for válido.

4.8 API RestFul

O acrônimo API que provém do inglês Application Programming Interface (Em português, significa Interface de Programação de Aplicações), trata-se de um conjunto de rotinas e padrões estabelecidos e documentados por uma aplicação A, para que outras aplicações consigam utilizar as funcionalidades desta aplicação A, sem precisar conhecer detalhes da implementação do software (Becode,2020).

Uma API RestFul realiza sua comunicação utilizando verbos HTTP realizando a troca de mensagens no formato JSON (*Javascript Object Notation*).

REST significa *Representational State Transfer.* Em português, Transferência de Estado Representacional. Trata-se de uma abstração da arquitetura da Web. Resumidamente, o REST consiste em princípios/regras/*constraints* que, quando seguidas, permitem a criação de um projeto com interfaces bem definidas. Desta forma, permitindo, por exemplo, que aplicações se comuniquem (Becode,2020).

De acordo com (Microsoft,2020) quando afirmamos que uma API é RestFul estamos afirmando que o princípio REST foi aplicado em cima do protocolo HTTP utilizando JSON.

Entre as vantagens de usar uma API RestFul está o ganho de velocidade por trafegar apenas informações reduzidas como o JSON, uso de memória e serialização.

4.9 Protótipo das Interfaces do Sistema

Os protótipos das telas e suas funcionalidades serão apresentadas nas subseções abaixo.

4.9.1 Tela de login

A figura 7 representa a tela de *login* com os campos de entrada com dados meramente ilustrativos para melhor entendimento do comportamento.

Figura 7 - Tela de *login*

Fonte: Elaborado pelo autor.

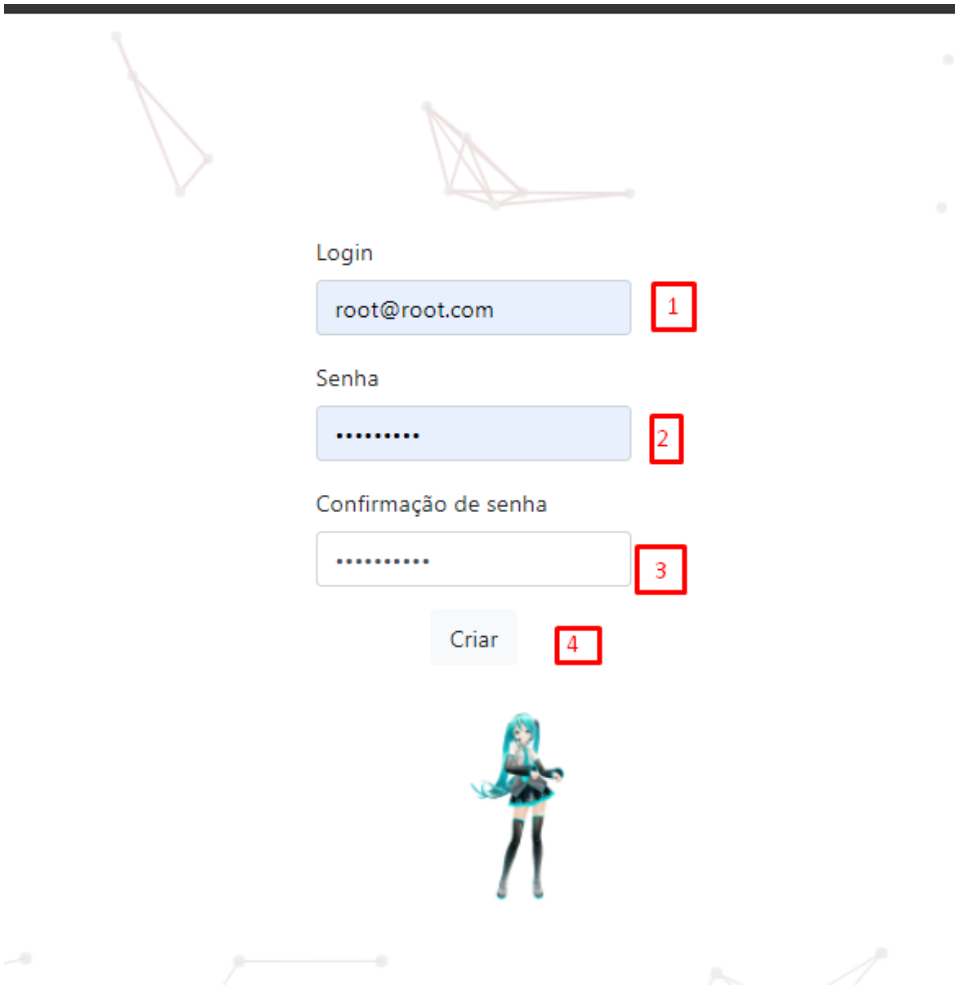
1. Entrada para login do usuário.
2. Entrada para senha do usuário.
3. Botão para realizar o login.

4. Cadastro de novo usuário.

4.9.2 Tela de Cadastro de usuários.

A figura 8 representa a tela de cadastro de usuário com os campos de entrada com dados meramente ilustrativos para melhor entendimento do comportamento.

Figura 8 - Tela de Cadastro de usuários.



Fonte: Elaborado pelo autor.

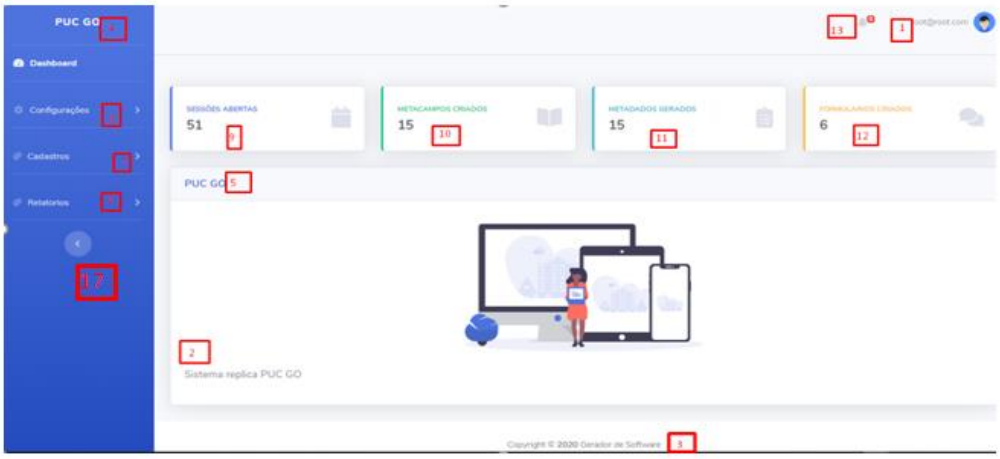
1. Entrada de login do usuário
2. Entrada de senha
3. Entrada de confirmação da senha

4. Botão para criar o usuário

4.9.3 Tela de inicial do aplicativo

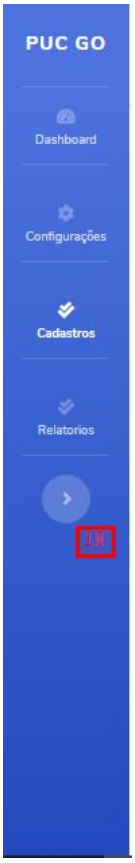
As figuras 9 a 14 representam a composição da tela de *dashboard* junto com os itens de menu.

Figura 9 – Tela de Home



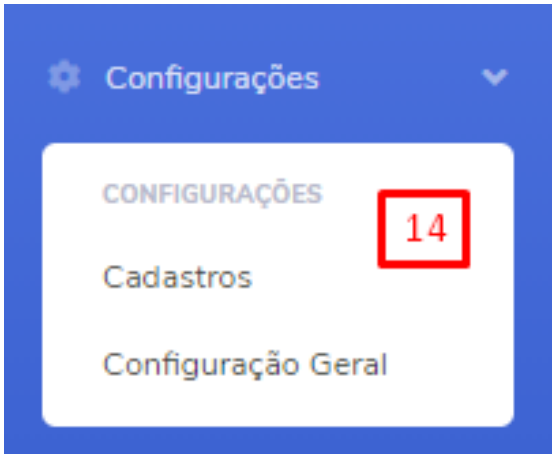
Fonte: Elaborado pelo autor.

Figura 10 – Menu lateral



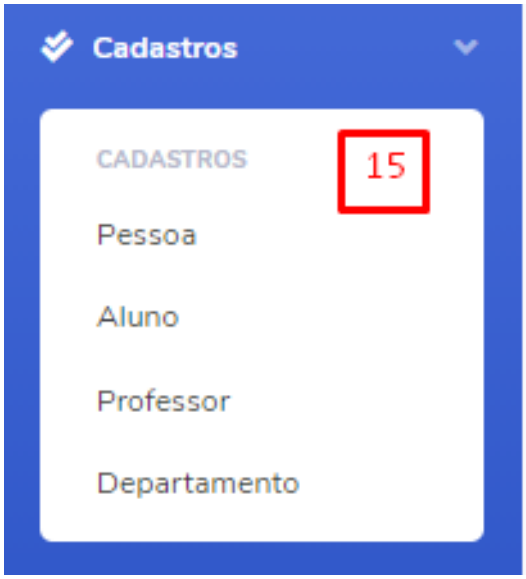
Fonte: Elaborado pelo autor.

Figura 11 – Configurações do sistema



Fonte: Elaborado pelo autor.

Figura 12 – Exempl de menu dinamico preenchido

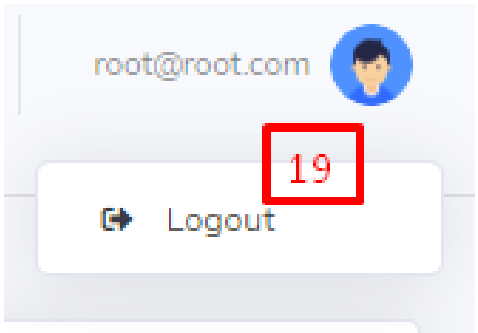


Fonte: Elaborado pelo autor.

Figura 13 – Relatório geral

Fonte: Elaborado pelo autor.

Figura 14 – Botão de sair do sistema



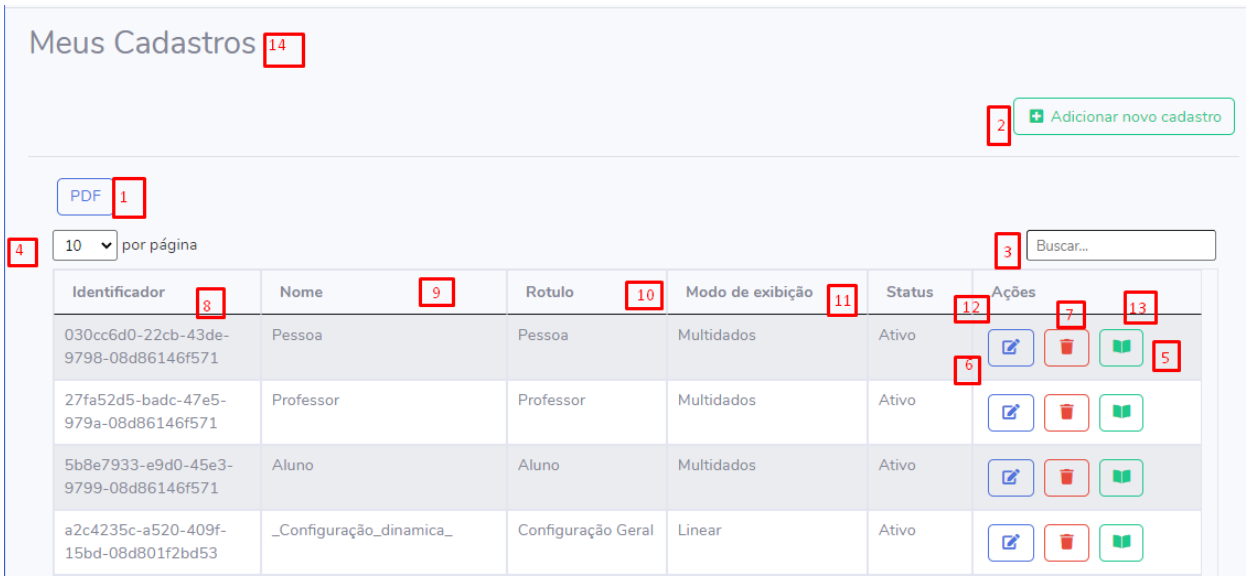
Fonte: Elaborado pelo autor.

1. Exibição do login.
2. Exibição do texto de sobre.
3. Exibição do texto de Rodapé.
4. Exibição do nome do sistema.
5. Exibição do nome do sistema.
6. Menu de configuração.
7. Menu de cadastro.
8. Menu de relatórios.
9. Exibição da quantidade de sessões abertas.
10. Exibição da quantidade de metacampos cadastrados.
11. Exibição da quantidade de metadados cadastrados.
12. Exibição da quantidade de metaformularios criados.
13. Exibição de alertas do sistema
14. Exibição das opções de configurações.
15. Exibição das opções de cadastros (com dados fictícios).
16. Exibição da opção de relatório
17. Botão para minimizar o menu.
18. Botão para maximizar o menu.
19. Opção para logout.

4.9.4 Tela de Listagem e de exclusão de metaformularios

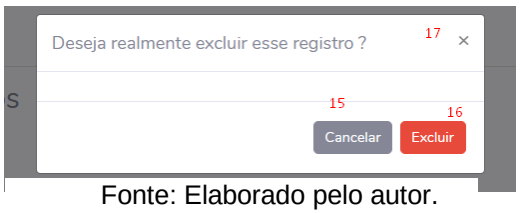
As figuras 15 e 16 mostra como é realizada a listagem e exclusão dos metaformularios.

Figura 15 - Tela de Cadastro de metaformulario



Fonte: Elaborado pelo autor.

Figura 16 - Caixa de confirmação



Fonte: Elaborado pelo autor.

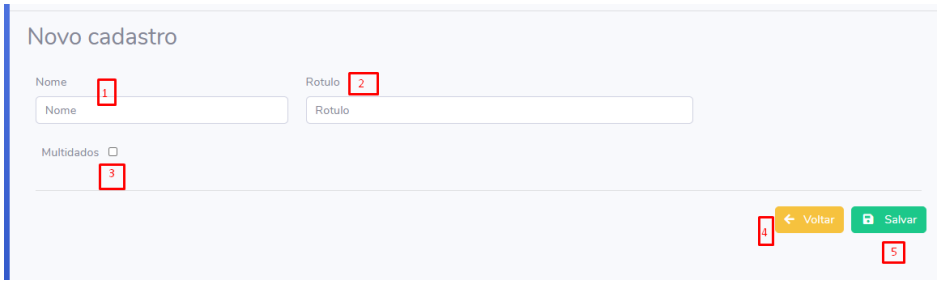
1. Botão para exportação da lista em formato PDF.
2. Botão para navegar para a página de criação de um novo metaformulario.
3. Campo texto para busca local na lista.
4. *DropDownList* para manipulação da paginação da lista.
5. Botão para navegar para a página de listagem/cadastro de metacampos.
6. Botão para navegar para a página de edição do metaformulario.
7. Botão de inativação do metaformulario.
8. Exibição do identificador do metaformulario.
9. Exibição do nome do metaformulario.
10. Exibição do rótulo do metaformulario.
11. Exibição do tipo do metaformulario.
12. Exibição do status do metaformulario.
13. Exibição das ações possíveis para o metaformulario listado.

14. Exibição do título da página.
15. Botão de fechar modal e cancelar exclusão.
16. Botão de confirmar exclusão.
17. Botão de fechar modal e cancelar exclusão.

4.9.5 Tela de Cadastro de metaformulario

A figura 17 mostra a tela de cadastro de um novo metaformulario.

Figura 17 - Tela de cadastro de metaformulario



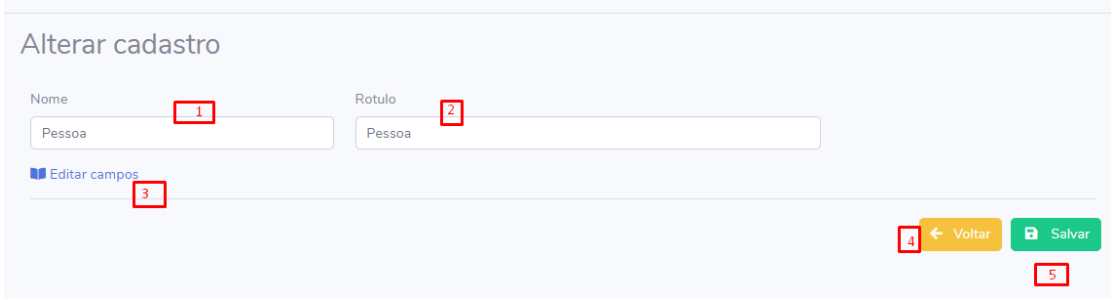
Fonte: Elaborado pelo autor.

1. Campo de exibição e alteração do nome do metaformulario.
2. Campo de exibição e alteração do rótulo do metaformulario.
3. *CheckBox* para definição de metacampo multidados ou linear.
4. Botão para voltar a tela anterior.
5. Botão para salvar as alterações.

4.9.6 Tela de edição de metaformulario

A figura 18 mostra a tela de alteração de um metaformulario.

Figura 18 - Tela de alteração de metaformulario



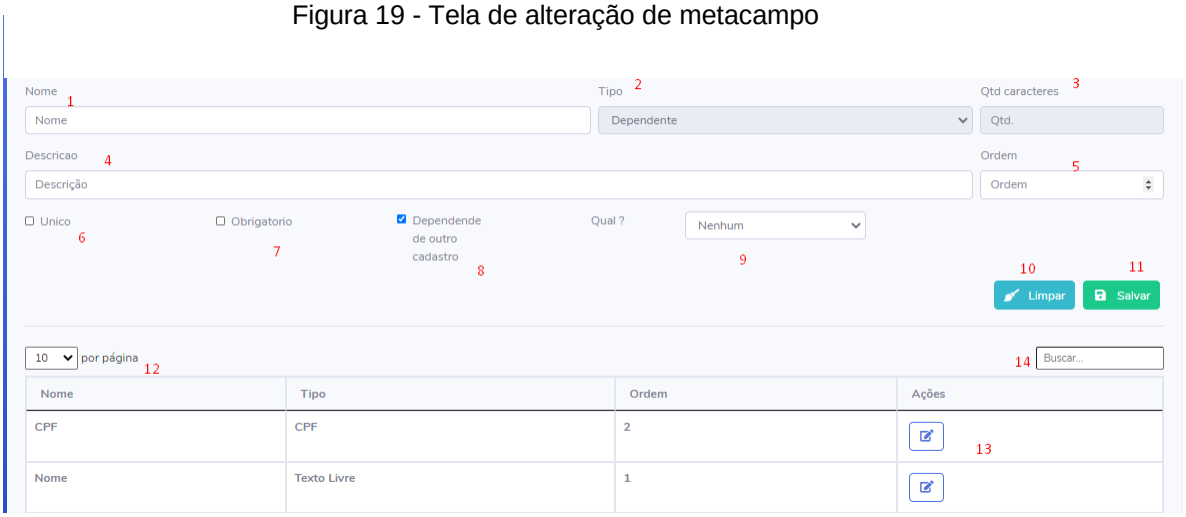
Fonte: Elaborado pelo autor.

1. Campo de exibição e alteração do nome do metaformulario.
2. Campo de exibição e alteração do rótulo do metaformulario.
3. Botão para edição de metacampos do metaformulario.
4. Botão para voltar a tela anterior.
5. Botão para salvar as alterações.

4.9.7 Tela de Cadastro de metacampo

A figura 19 está presente a tela de listagem de metaformularios que possui também a responsabilidade de inclusão e edição dos metacampos.

Figura 19 - Tela de alteração de metacampo



Fonte: Elaborado pelo autor.

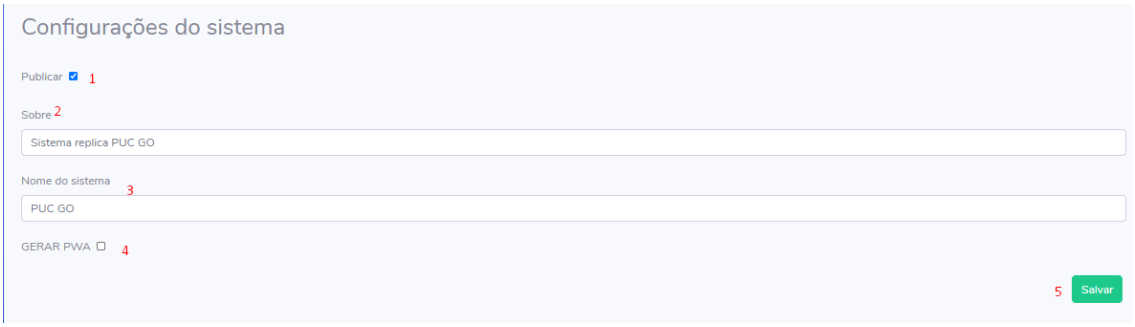
1. Campo de exibição e alteração do nome do metacampo.
2. Campo de exibição e alteração do tipo do metacampo.
3. Campo de exibição e alteração da quantidade de caracteres do metacampo.
4. Campo de exibição e alteração da descrição do metacampo.
5. Campo de exibição e alteração da ordem de exibição do metacampo.
6. Campo *checkbox* para informar se o campo é único.
7. Campo *checkbox* para informar se o campo é obrigatório.
8. Campo *checkbox* para informar se o campo é dependente.

9. Campo *dropdownlist* para informar qual formulário o metacampo pertence caso ele seja dependente.
10. Botão limpar.
11. Botão salvar.
12. Paginação da lista.
13. Botão de edição de um metacampo.
14. Campo de busca local na lista do metacampo.

4.9.8 Tela de Configuração do sistema

A figura 20 apresenta a tela de configuração do sistema que é feita com os próprios recursos do sistema.

Figura 20 - Tela de configuração do sistema



Fonte: Elaborado pelo autor.

1. *Checkbox* para realização do CI/CD.
2. Campo de exibição e alteração do texto de sobre do sistema.
3. Campo de exibição e alteração do nome do sistema.
4. *Checkbox* para permissão de geração do PWA.

4.9.9 Tela de Listagem de metadados e exclusão

A figura 21 e 22 apresenta como é a listagem e a exclusão dos metadados de um metaformulario com dados fictícios para facilitar o entendimento.

Figura 21 - Tela de listagem de metadados

Fonte: Elaborado pelo autor.

Figura 22 - Tela de confirmação de exclusão de uma

Fonte: Elaborado pelo autor.

1. Campo de exibição do título do cadastro.
2. Campo de exportação da listagem em formato pdf.
3. Botão de chamada da tela de novo cadastro de uma tupla.
4. Campo de busca na lista de metadados.
5. *DropDownList* de paginação.
6. Metacampos dos metadados presentes.
7. Botão da chamada da tela de alteração da tupla.
8. Botão de exclusão da tupla.
9. Botão de cancelar a exclusão.
10. Botão de confirmar a exclusão de uma tupla.

4.9.10 Tela de Cadastro de metadados

A figura 23 representa o formato da tela de cadastro e alteração de uma tupla com dados fictícios.

Figura 23 - Tela de cadastro e alteração de uma tupla.

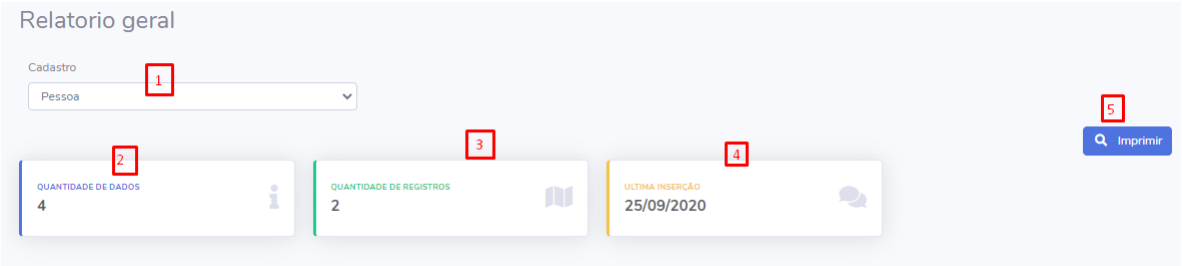
Fonte: Elaborado pelo autor.

1. Campo de entrada e saida de um metacampo para seu metadado.
2. Campo de entrada e saida de um metacampo para seu metadado.
3. Botão de fechar.
4. Botão de salvar o cadastro/alteração.

4.9.11 Tela de Relatório geral

A figura 24 mostra como é a tela de relatório geral.

Figura 24 - Tela de relatório geral



Fonte: Elaborado pelo autor.

1. *DropDownList* de entrada do metaformulario.
2. Exibição da quantidade de dados do metaformulario.
3. Exibição da quantidade de registro do metaformulario.
4. Exibição da data da última alteração do metaformulario.
5. Botão de impressão dos resultados do relatório.

5 Considerações finais

No início deste projeto foi analisado que a automatização de processos se faz presente em diversas áreas do desenvolvimento de *software,* porém ainda é pouco explorada quando se trata da geração de softwares, e em alguns casos complexas quando se trata de realização de uma entrega continua em ambiente produtivo.

Diante disso, o trabalho apresentado teve como objetivo a criação de um sistema criador de *software* com a finalidade de auxiliar usuários de alto ou baixo nível técnico a desenvolverem *softwares* de maneira simples, segura e escalável.

Para que isso fosse possível foi necessário o levantamento de requisitos, levando em conta as principais necessidades de um *software* e o que todos os *softwares* têm em comum para que pudéssemos ter um ferramental para abranger vários cenarios do desenvolvimento de software. Diante disso, foi levantado os requisitos de usuários, funcionais, e de qualidade do sistema, especificações dos casos de usos, estudos de tecnologias como *.Net Core, Asp.Net Core MVC, C#, EFC e* API RestFul e a realização de testes.

O sistema foi divido em dois níveis: Usuário e *Root*: o primeiro tem apenas permissões de manipular o que chamamos nesse projeto de metadados. O segundo além de poder manipular metadados pode manipular metacampos e metaformularios e tem acesso livre a qualquer outra mudança que ele achar necessário no sistema.

Após teste no sistema e validações, o Gerador de Software se mostrou eficaz para a criação de sistemas, controle de informações para inclusão, alteração, exclusão e visualização, relacionamentos entre dados e entrega continua, se mostrando eficaz em ter disponibilidade em outros ambiente além da WEB, como os ambientes *Desktop* e *mobile.*

5.1.1 Sugestões de trabalhos futuros

Pensando em uma solução mais abrangente aos usuários do sistema, como a implementação de um modulo de realização de vendas e controle de estoque junto a integração com empresas de pagamento *online* e uma política de cópia de segurança e restauração de dados poderia ser um tema abordado em trabalhos futuros.

6 Referências

Microsoft. Microsoft, 2020. A central de documentação e aprendizado da Microsoft para desenvolvedores e profissionais de tecnologia. Disponível em: https://docs.microsoft.com/pt-br. Acesso em 25 nov. 2020.

Alves, Thiago. *Entity Framework code-first.* Devmedia, 2013. Disponível em: https://www.devmedia.com.br/entity-framework-code-first/29705. Acesso em 25 nov. 2020.

Pires, Jackson. O que é API? REST e RESTful? Conheça as definições e diferenças, 2016. Disponível em: https://becode.com.br/o-que-e-api-rest-e-restful/.Acesso em: 23 nov. 2020.

Toledo, Ricardo. Geração de valor pelo uso de TI. Fundação Getúlio Vargas, São Paulo, p 27,2020.

Dos Santos, I. B.; Sandman, A.; De Souza, B. E.; Pizarro Schmidt, C. A.; Marcolin, J. F.; Melges, a. i. Automatização de processos rurais: Proposta de implementação de um *gateway* de internet das coisas (iot) para simplificar a automação da aquicultura. The *Journal of Engineering and Exact Sciences*, *[S. l.]*, v. 6, n. 1, p. 0001-0007, 2020. DOI: 10.18540/jcecvl6iss1pp0001-0007. Disponível em: https://periodicos.ufv.br/jcec/article/view/9356. Acesso em: 26 nov. 2020.

Gonçalves, et al. IEEE Std 830 Prática Recomendada Para Especificações de Exigências de Software. Universidade de Lisboa, Lisboa,2004.

Groffe, Renato. Modelagem de sistemas através de UML: uma visão geral. Devmedia, 2013. Disponível em: https://www.devmedia.com.br/modelagem-de-sistemas-atraves-de-uml-uma-visao-geral/27913. Acesso em 25 nov. 2020.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Murilo Barros Peixoto
do Curso de Ciência da computação ,matricula 2015.2.0028.0040-1 ,
telefone:62 9 9908-2008 e-mail murilombp2214@gmail.com , na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Desenvolvimento do Gerador de Software
, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 11 de dezembro de 2020 .

Assinatura do(s) autor(es): Murilo B. Peixoto

Nome completo do autor: Murilo Barros Peixoto

Assinatura do professor-orientador:

Nome completo do professor-orientador: Eugênio Júlio Messala Cândido Carvalho